

Manu Devos

*LE LIVRE
DU
DISQUE M.S.X.*



BCM

LE LIVRE DU DISQUE M.S.X.

par
MANU DEVOS

BCM 1987

Le Livre du VIC (épais)
Le Livre du 64
Le Livre du MSX
Les Dessous du SPECTRAVIDEO
Le Livre de l'AMSTRAD
Le Livre du MS/PC-DOS pour PC et AT
Ecrire en dBASE II et III
Le Maître de l'eu sur C-4, C128 (T1)

par B. MICHEL
par B. MICHEL
par D. MARTIN
par D. MARTIN
par F. PIETTE
par C. MICHEL
par F. BRUNET

DISQUETTES DISTRIBUEES PAR B.C.M. s.r.l.

Ces disquettes contiennent les programmes des livres concernés

Le Disque du 64
Le Disque de MS/PC-DOS
Le Disque de dBASE

Le Disque du MSX
Le Disque de l'AMSTRAD
Le Disque de QuickBASIC

EN PREPARATION

Le Livre du MSX-2
Le Livre du FICKE
Explainer TurboRisc
PC, XT, AT, Dépannage, Maintenance et Améliorations
Les Secrets d'UNIX
Les Secrets de PageMaker
Les Secrets d'AutoCAD

par D. MARTIN et G. GAVAGE
par D. MARTIN et B. MICHEL
par V. LABAYE et A. RIGO
par B. MICHEL et M. BENOIT
par G. HERTZ et V. LABAYE
par A.-L. SURY
par P. LECOCQ

1ère édition Septembre 1987
Imprimé en Belgique par BONTON - 4600 CHENEIE
Dépot légal : D/1987/0827/1

Copyright B.C.M. s.r.l.

24, route de la Sapinière, B - 4960 BANNEUX, BELGIQUE
ISBN : 7-3713-010-7

Toute reproduction, sans réserve, l'usage ou l'emploi d'un extrait quelconque de ce livre par quelque procédé que ce soit, est formellement interdite de l'éditeur.

Distribut. en France par J.C.Y. Diffusion, B.P. 96, F-77042 LAURY/MARNE Cedex
en Belgique par PHILIPS, 1 Place de Bruxelles, 1050 BRUXELLES

1. Les disques du système MSX	9
1.1 Capacité en unités de disquettes	9
1.2 Constitution physique d'une disquette	9
1.3 Les différents types de disquettes	10
1.4 Le formatage d'une disquette	10
2. Les noms des unités de disquettes et des fichiers	13
2.1 Les noms des unités de disquettes	13
2.2 Les noms des fichiers	13
2.2.1 Qu'est-ce qu'un fichier	13
2.2.2 Le fichier séquentiel	13
2.2.3 Le fichier à accès direct	13
2.2.4 La structure du nom du fichier	16
2.2.5 Les extensions réservées	17
2.2.6 Les fichiers périphériques	18
2.2.7 Les caractères de substitution	18
3. Le Disk-Basic	21
3.1 Les commandes de gestion des fichiers	22
3.1.1 SAVE	22
3.1.2 LOAD	23
3.1.3 ERASE	24
3.1.4 BLOAD	25
3.1.5 MERGE	26
3.1.6 NAME	27
3.1.7 KILL	28
3.1.8 FILES - LFILES	29
3.1.9 COPY	30
3.1.10 RUV	32
3.1.11 MAXFILES	32
3.2 Les CALL du Disk-Basic	34
3.2.1 CALL FORMAT	34
3.2.2 CALL SYSTEM	34
3.3 La manipulation d'un fichier séquentiel	35
3.3.1 L'ouverture d'un fichier séquentiel	36
3.3.2 L'écriture dans un fichier séquentiel	37
3.3.3 La lecture d'un fichier séquentiel	40
3.3.4 Le fermeture d'un fichier séquentiel	41
3.3.5 Programme exemple de fichier séquentiel	42
3.3.5.1 Création ou extension d'un fichier	42
3.3.5.2 Construction d'un fichier	43
3.3.6 TIME INPUT#	44
3.4 Les fonctions spécifiques aux fichiers séquentiels	45
3.4.1 INPUT#	45
3.4.2 EOF	45
3.4.3 EOC	46
3.4.4 LOP	47
3.5 Les manipulations d'un fichier à accès direct	48
3.5.1 L'ouverture d'un fichier à accès direct	49
3.5.2 FIELD - Création de nom	50

Les disques du système MSX

7.5.42 29 Fonction 29 non-suppétée	187
7.5.43 30 Gec Data	188
7.5.44 31 Sec Data	188
7.5.45 32 Gec Time	189
7.5.46 33 Sat Time	190
7.5.47 34 Set/Ready Verify Flag	190
7.5.48 35 Absolute Disk Read	191
7.5.49 36 Absolute Disk Write	192
7.6 Direct Bios Access	198

8. L'éditeur MSX-DOS et les fichiers batch	197
--	-----

8.1 L'éditeur du MSX-Dos	197
8.1.1 Généralités	197
8.1.2 Les caractères de contrôle	200
8.2 Les fichiers de commande ou Batch Files	201
8.2.1 Généralités	201
8.2.2 Le fichier AUTOEXEC.BAT	205
8.2.3 Les paramètres d'un fichier de commande	206

ANNEXE : Les différents types de disques	209
--	-----

1. Les Disques de 5 1/2	209
2. Les Disques de 5 1/8	210

1.1 Capacité en unités de disquettes

Un ordinateur MSX peut supporter au maximum 8 lecteurs de disquettes. En fait cela dépend beaucoup de votre appareil, car on ne connecte pas l'unité directement à l'ordinateur mais via une interface, le contrôleur disque.

Ce contrôleur disque peut gérer deux lecteurs de disquette au maximum. Il contient une ROM de 1K qui se charge de ce travail : il doit nécessairement se loger dans un port cartouche ou slot. Le standard MSX a défini un maximum de 4 slots. Le slot 0, qui contient les ROM Bios et Basic, est en fait relié à l'initiateur de votre MSX, donc non accessible pour l'utilisateur. Selon le marque de votre ordinateur, vous pouvez trouver jusqu'à 3 slots d'extension que vous utiliserez habituellement pour vos cartouches de jeux). Si vous y glissez des contrôleurs disques, vous disposerez d'un maximum de 3 contrôleurs disques externes, plus un éventuel contrôleur interne sur les ordinateurs avec un lecteur intégré. Cela nous donne donc une capacité théorique de 4 contrôleurs, soit 8 unités de disquettes (voir le chapitre 9).

1.2 Constitution physique d'une disquette

Une disquette est un plateau de forme circulaire recouvert d'une couche d'oxyde de fer sensible aux champs magnétiques. Il ne s'agit pas, comme pour nos disques 33T, d'une gravure mécanique d'un disque de plastique : la disquette rassemble plus à la bande magnétique de nos enregistreurs qu'à un 33T.

L'écriture et la lecture des informations sur une disquette s'effectue par rotation de la disquette devant une tête de lecture/écriture tout comme la bande défilait devant la tête de votre enregistreur. La vitesse de rotation du lecteur de disquettes est de 300 t/min, ce qui correspondrait à une vitesse de défilement de 60 à 100 cm/sec pour un enregistreur qui aurait les mêmes performances que votre unité de disquettes.

Une autre différence est que la tête de lecture/écriture ne suit pas un sillon comme dans un 33T, mais doit se positionner sur une piste circulaire grâce à un mécanisme de déplacement. Il y a ainsi une série de pistes concentriques sur votre disquette dont le nombre dépend du type de support. On rencontre des lecteurs de 40 pistes (appelés SD

- Single Density) et des lectures de 80 pistes (appelées DD - Double Density).

En général, les lecteurs MSX ont 80 pistes. On numérote les pistes de 0 à 79, la piste 0 étant la plus éloignée du centre de la disquette. Le plateau recouvert d'oxyde de fer est entaillé dans un boîtier plastique rigide pour les disquettes de 3 1/2 et dans une jaquette souple pour les disquettes de 5 1/4.

Une piste est divisée en secteurs. Les secteurs sont des portions de piste de contenance égale en nombre de caractères stockés (512). Le secteur est l'unité de lecture ou d'écriture sur la disquette; autrement dit, c'est la plus petite information que le hardware puisse lire ou écrire sur une disquette en une seule opération. On numérote aussi les secteurs en donnant le numéro 0 au premier secteur de la piste 0 et au dernier numéro au dernier secteur de la dernière piste. Vous devez encore savoir que certains lecteurs de disquettes sont équipés de deux têtes de lecture/écriture, une sur chaque face de la surface magnétique. On parle dès lors de lecteur Simple Face ou Double Face (Single Side - Double Side).

Deux méthodes d'enregistrement électronique des informations appelées FM, pour Frequency Modulation, et MFM, pour Modified Frequency Modulation sont utilisées. Les contrôleurs MSX emploient la MFM qui permet de stocker deux fois plus d'informations que la méthode FM.

1.3 Les différents types de disquettes

Cheque type de disquette a reçu un code. La norme MSX a désté jusqu'à présent huit types de disquettes dont la liste suit.

CODE	CAPACITÉ	FORMAT	FACE	PISTE	SECTEUR
FD	362496	3 1/2	Simple	80	9
FD	730112	3 1/2	Double	80	9
FD	322560	3 1/2	Simple	80	8
FD	649216	3 1/2	Double	80	8
FD	179712	5 1/4	Simple	40	9
FD	362496	5 1/4	Double	40	9
FD	140256	5 1/4	Simple	40	8
FD	322560	5 1/4	Double	40	8

Attention! tous les contrôleurs-disques ne supportent pas tous les types de disquettes. C'est ainsi que les contrôleurs les plus récents ne supportent plus les disquettes de type FD à FF.

1.4 Le formatage

Une disquette neuve ne peut pas être employée directement

pour sauvegarder des programmes ou des fichiers. En effet, bien qu'elle contienne déjà une ou deux faces magnétisables, que les 40 ou 80 pistes soient bien présentes, la disquette neuve n'est pas encore "découpée" en secteurs. C'est intentionnellement que le fabricant s'a laissé dans cet état initial, chaque système pourra découper la disquette suivant ses besoins. On rencontre des formats où chaque piste contient 25 secteurs de 128 octets, ou encore 10 secteurs de 512 octets, ou, comme en MSX, 8 ou 9 secteurs de 512 octets (format identique aux IBM PC).

L'opération de formatage est réalisée conjointement par un programme spécifique et par le hardware. Elle consiste à écrire sur la piste, pour identifier le début et la fin de chaque secteur, leur ordre, leur longueur, la piste où ils se trouvent ainsi que le numéro de la tête qui se voit passer (voir le chapitre 4).

Si vous êtes novice dans l'emploi des disquettes, cherchez une fois parvez formater une disquette "terge par un CALL FORMAT (ou FORMATT, sous BASIC ou FORMAT, sous DOS (quand le signe A) apparaît au début de l'ligne. Attention, cette commande efface complètement le contenu de la disquette. Si vous en formatez une qui contenait déjà des fichiers,

Les noms des unités de disquettes et des fichiers

2.1 Les noms des unités de disquettes

En Disk-Basic et en MSX-DOS, les huit unités de disquette connectables à l'ordinateur portent chacune un nom pour la facilité d'utilisation.

Ce nom est en fait une simple lettre, de A à H, suivie du caractère ":", l'unité A: est le premier lecteur de votre système, l'unité B: le second, et ainsi de suite jusqu'au lecteur H:. Si vous n'avez qu'une unité, elle portera donc le nom A:.

Dans ce cas, comment réaliser des copies de disquettes? Heureusement la norme MSX a prévu une solution à ce problème. En effet, n'ayant pas détecté de seconde unité de disquettes, soit parce qu'elle n'existe pas, soit parce qu'elle n'était pas allumée au moment du démarrage du système, celui-ci va considérer cependant que vous avez un pseudo deuxième lecteur qui portera effectivement le nom B:.

Lorsque vous demanderez une opération avec l'unité de disquette B:, le système va vous inviter à insérer la disquette présente dans votre unique lecteur et à insérer celle qui serait présente dans votre second lecteur si vous en avez un. Le message est le suivant:
Insert diskette for drive B:
and strike a key when ready

ce qui signifie littéralement: "insérez la disquette dans l'unité de disquette B: et enfoncez une touche lorsque vous êtes prêt".

A partir de ce moment, votre unique lecteur est devenu le lecteur B:. De même, lorsque vous voudrez revenir au lecteur A:, le système produira le même message mais en vous invitant cette fois à insérer la disquette dans le lecteur A:.

Si vous avez quatre unités, déterminer quel lecteur porte le nom A:, B:, C:, ou D: est un peu plus compliqué; nous devons envisager trois éventualités.

1. Les unités de disquettes sont toutes sous tension à l'initialisation du système MSX;

Si vous allumez l'ordinateur normalement (c-à-d, allumer d'abord toutes les unités de disquettes et puis l'ordinateur), les unités A: et B: seront celles connectées au contrôleur disque se trouvant dans le slot de numéro le plus bas. L'unité A: sera celle dont le sélecteur interne est sur A, l'unité B: celle dont le sélecteur interne est sur B. Quand aux unités C: et D:

ainsi l'aaa ou l'ordre directement l'enregistrement numéro 27 ou 456 sans avoir l'aaa tous les prétendant aaaa n'est pas l'aaa avec un fichier séquentiel.

En plus de la longueur de l'enregistrement, il faut aussi prévoir d'avance une répartition de cette longueur entre les différents mots de l'enregistrement. Par exemple, dans un fichier d'adresse, 20 caractères pour la zone, 15 caractères pour le prénom, 10 caractères pour la rue, 4 caractères pour le numéro, et ainsi de suite pour chaque zone.

Il faudra aussi employer une technique spéciale pour tester les informations dans chaque zone de l'enregistrement, ainsi pour les zones alphabétiques, on pourra les aligner à gauche ou à droite et tester les positions inutilisées de la zone avec des espaces grâce à une instruction Basic spéciale. De même pour les zones numériques, il faudra les aligner à droite et les tester avec une instruction différente suivant qu'il s'agit d'une variable entière, simple précision ou double précision.

2.2.1 Le structure du nom du fichier

Chaque fichier dans le dossier MSK porte un nom. Ce nom est divisé en deux parties: d'une part le nom du fichier proprement dit et d'autre part son extension ou son type. Ces deux parties sont séparées par le signe '.',

Exemple: PROGRAM.BAS
MSKDOS.BYS
LETRES.TXT
PACMAN.ASC
JEU.I
MENU
ADRESSES.MSK

et le nom du fichier proprement dit ne peut dépasser 8 caractères.

Et la zone de fichier ne doit pas nécessairement être suivie d'une extension. Dans ce cas, on ne met rien de plus.

et L'extension ne peut dépasser 3 caractères.

et Si vous incluez des minuscules dans le nom du fichier ou dans son extension, elles seront automatiquement converties en majuscules.

et Vous pouvez employer n'importe quel code y compris les caractères et les codes graphiques ainsi que l'exclusion des caractères suivants: ! " # \$ % & ' () * + , - . / : ; [\] ^ _ ` { | } ~

Les exemples ci-dessous montrent bien l'emploi de l'extension.

PROGRAM.BAS L'extension .BAS signifie que le fichier PROGRAM est en Basic.

LETRES.TXT L'extension .TXT signifie que le fichier

LETRES est du texte plein et n'a pas de programme Basic.

PACMAN.ASC L'extension .ASC signifie que le programme PACMAN est sauve en ASCII plutôt qu'en binaire.

MSKDOS.BYS L'extension .BYS signifie que le fichier MSKDOS fait partie du système.

JEU.I L'extension .I signifie probablement qu'il s'agit d'un jeu vidéo.

MENU On voit ici que l'extension n'est pas obligatoire.

ADRESSES.MSK L'extension peut aussi servir à indiquer un type de fichier ou un langage.

2.2.2 Les extensions réservées

Vous avez libre choix de nom et de l'extension pour vos fichiers, cependant certaines extensions sont réservées pour le MSKDOS et d'autres pour certains logiciels comme les Assembleurs-Éditeurs. De plus, vous pouvez choisir de ne pas employer les noms d'extension et laisser le système les reconnaître de lui-même.

- .COM Fichier en langage machine (compilé) de programme d'un ordinateur le MSKDOS. Il se charge de l'interpréter de la source.
- .BAT Fichier batch le MSKDOS, il s'agit d'une séquence de commandes MSKDOS en ASCII.
- .BAS Fichier programme en BASIC écrit en langage source.
- .FOR Fichier programme en langage FORTRAN.
- .CPR Fichier programme en langage COBOL.
- .PAS Fichier programme en langage Pascal.
- .PLI Fichier programme en langage PLI.
- .C Fichier programme en langage C.
- .ASM Fichier programme en langage Assembleur (Z80).
- .MER Fichier programme en langage Merge.
- .MAC Fichier source de Macro-Assembleur.
- .REL Fichier langage Relais-Programmeur.
- .BWK Fichier langage de travail de l'utilisateur.
- .SYS Fichier de système d'exploitation DOS.
- .TXT Fichier de Texte.
- .ASC Fichier programme en ASCII écrit en ASCII.
- .PIC Fichier programme en langage graphique (PICUREL).
- .LIB Fichier bibliothèque (Contient des routines de programme).
- .TMP Fichier temporaire en CP/M.
- .TMP Fichier temporaire en MSKDOS (IBM).

Certains noms de fichiers sont réservés par le système d'exploitation ou par le Basic, il ne faut donc pas les utiliser.

MSKDOS.BYS fichier d'initialisation du MSKDOS.

COMMAND.COM Fichier qui contient les commandes
nécessaires au MSXDOS.

AUTOCHECK.BAT Fichier contenant une série de commandes
MSXDOS qui sera auto-exécutée à l'allumage de
l'ordinateur. Vous pouvez le créer ou modifier son
contenu selon vos besoins.

AUTOCHECK.BAS Fichier contenant un programme Basic qui
sera auto-exécuté lors du premier passage en Basic ou à
l'activation de l'ordinateur si votre disque ne contient
pas le fichier MSXDOS.SYS. Vous pouvez le créer ou
modifier son contenu suivant vos besoins.

De plus, si vous complez exécuter vos fichiers sur un PC
IBM, nous vous recommandons de pas employer les noms de
fichiers suivants :

IO.SYS, MSDOS.SYS, ANSI.SYS, VOICE.SYS, CONFIG.SYS

2.2.4 Les Fichiers périphériques.

Il existe cinq noms de fichiers tout à fait particuliers.
Plus qu'à désigner un fichier sur le disque, ils
désignent des appareils périphériques qui, grâce à ces noms
spéciaux, apparaîtront au système d'exploitation comme de
simples fichiers disques. Ces noms de fichiers désignent
l'imprimante, le clavier, l'écran et une entrée/sortie
auxiliaire telles que l'interface RS-232. Ces noms sont :

LSI, PRN, CON, NUL et AUX

LSI et PRN désignent tous deux l'imprimante comme
destination des données. Ainsi, vous pouvez, par le choix du
nom de votre fichier, diriger vos données à un vrai fichier
disque ou vers l'imprimante.

CON (Console) désigne l'écran comme destination de vos
données ou le clavier comme source de vos données. Dans ce
dernier cas, l'entrée par le clavier se termine par une
marque de fin de fichier que vous devrez poser en frappant
CTRL-Z suivi de RETOUR.

NUL correspond à un fichier nul. Les données sont simplement
écartées plutôt que d'entrer dans un fichier.

AUX désigne vos données à un périphérique externe ou utilise
ce périphérique externe comme source de données. Ce
périphérique peut être par exemple une interface RS-232 de
télécommunication connectée via un Modem à une ligne
téléphonique.

2.2.7 Les caractères de substitution

Pour rechercher un nom de programme dont on ne se rappelle
que l'extension (.BAS), il serait souhaitable de ne pouvoir
afficher que les programmes ayant ce type d'extension, de
même qu'il serait intéressant de rechercher tous les

fichiers s'appellent 'BUDGET', par exemple, quels qu'en
soient les extensions.

A cet effet, nous disposons d'un premier caractère de
substitution qui est le caractère aléatoire. Il remplace une
série de caractères quelconques. Son alias se lit à la
première partie ou à l'extension du nom de fichier, mais pas
aux deux en même temps.

Ainsi, la commande Basic FILES '*.*.BAS' visualisera tous les
fichiers ayant l'extension .BAS. La commande Basic FILES
'BUDGET.*' visualisera tous les fichiers s'appelant 'BUDGET',
par exemple BUDGET.JAN, BUDGET.FEV, BUDGET.MAR,
BUDGET.AVR, ..

Bien entendu, on peut aussi utiliser l'antérieur des deux
codes du nom d'un fichier comme c'est le cas dans la
commande FILES '*.*' qui visualisera absolument tous les
fichiers de votre disque (la commande FILES sans
paramètres produit le même effet).

Le deuxième caractère de substitution est le caractère
'point d'interrogation' '?'. Contrairement à l'antérieur,
il remplace un seul et unique caractère du nom du fichier ou
de son extension. Ainsi la commande FILES 'JEU?.BAS'
affichera les fichiers JEU1.BAS, JEU2.BAS, JEU3.BAS,
JEU4.BAS et de tels fichiers existent sur le disque bien
entendu.

On peut employer plusieurs codes de substitution dans un nom
de fichier comme dans la commande FILES 'JEU?.*.?' qui
affichera les fichiers JEU1.BAS, JEU2.BAS, JEU3.BAS,
JEU4.BAS, JEUX.OBJ, JEUX.TXT, JEUX.COM.

Chapitre 3

Le Disk-Basic

Le Disk-Basic est une extension des commandes, instructions et fonctions du MSX-Basic pour couvrir tous les aspects de la manipulation des fichiers sur disquette.

Le Disk-Basic devient actif dès qu'une interface-disque a été insérée dans un des slots de votre MSX et que le démarrage de l'ordinateur s'effectue SANS enfoncer la touche SHIFT (= majuscules) et en l'absence de disquette dans le lecteur A1. Si une disquette est insérée dans le lecteur A1, le système ne sélectionnera le Disk-Basic que si cette disquette NE contient PAS le fichier MSXDDS.SYS.

Dans ce chapitre, nous ne décrirons que les mots spécifiques au Disk-Basic, en supposant que le MSX-Basic est connu. L'ordre de présentation est uniquement dicté par des considérations d'ordre didactique. Référez-vous à la table des matières pour trouver rapidement la syntaxe d'un mot-clef particulier.

La notation de la syntaxe est standard. En voici un court résumé et, à titre d'exemple, prenons l'instruction INPUT.

```
INPUT ["<MESSAGE>";] [ <VARIABLE> [,VAR2,VAR3,...,VARn]
```

Les crochets signifient que tout ce qui s'y trouve inclus est optionnel. Ainsi dans l'exemple ci-dessus, le message, les guillemets et le point virgule ne sont pas obligatoires.

Les symboles < > signifient que le texte qu'ils encadrent doit être fourni par vous.

Vous verrez également apparaître les symboles suivants:

<nom fichier> Ce qui se trouve entre les symboles <...> doit être un nom de fichier valide sous la forme d'une chaîne.

<spéc. fic.> Ce qui se trouve entre les symboles <...> doit être une spécification de fichier c'-a-d. un nom d'unité de disquette et un nom de fichier, (A1:ESSA1.BAS), par exemple. Le nom de l'unité n'est pas obligatoire. Le système cherchera alors le nom du fichier dans l'unité dernièrement spécifiée. La spécification de fichier peut être donnée par une constante chaîne ou une variable chaîne.

3.1 Les commandes de gestion de fichiers

3.1.1 SAVE

SAVE <spéc. fichier> (,A)

Cette instruction est équivalente à l'instruction OSAVE du BASIC normal. Elle permet de sauvegarder un programme BASIC résident en mémoire sur un fichier disque. L'option A permet de sauvegarder le fichier en ASCII, autrement le fichier est sauvegardé en format binaire compressé. Voir le chapitre 4 pour les différents formats.

<spéc. fichier> est une chaîne de caractères ou une variable qui spécifie le nom de l'unité et le nom du fichier. Le nom de l'unité, le nom du fichier et l'extension doivent se conformer aux règles décrites dans les chapitres 2.1 et 2.2.4.

Si le fichier existe déjà sur le disque, son contenu sera remplacé par le programme se trouvant en mémoire. Le format ASCII prend plus de place sur le disque, mais permet une lecture de ce fichier par les instructions sans erreur en BASIC du même type au MSX-DOS qui veulent que ce fichier soit en ASCII.

Attention! Pour sauvegarder le programme en mémoire, vous utilisez les instructions OSAVE et SAVE pour sauvegarder le programme respectivement en binaire compressé et en ASCII. Mais pour le disque, les instructions deviennent SAVE et SAVE,,,A pour le mode ASCII.

Le nom du fichier ne peut pas contenir de caractères de substitution, et si le nom de chaque unité est spécifié, le programme sera sauvegardé sur le lecteur couramment sélectionné.

SAVE"PROGRAM2.BAS" Sauve le programme résident en mémoire sous le nom "PROGRAM2.BAS" sur l'unité couramment sélectionnée.

SAVE"0:JEU.001" Sauve le programme résident en mémoire sous le nom "JEU.001" sur l'unité 0.

SAVE"A:JEU.002",A Sauve le programme résident en mémoire sous le nom "JEU.002" sur l'unité A; en format ASCII.

IO SAVE"JEU.003" La commande SAVE est aussi une instruction et peut donc être intégrée à un programme.

IO P0="JEU.004" On peut également extraire le nom du programme dans une variable.

20 SAVE P0 On peut également sauvegarder un programme sur le fichier périphérique spécial AUX mais cette opération est réservée à ceux qui disposent d'une interface RS232 (voir chapitre 2.4).

SAVE "AUX",A Sauver le programme dans un autre

ordinateur ou sur une autre carte de mémoire RS232.

3.1.2 LOAD

LOAD <spéc. fichier> (,A)

Cette instruction permet de charger en mémoire un fichier contenant un programme BASIC. Ce fichier peut avoir été sauvegardé en format binaire compressé ou en ASCII.

<spéc. fichier> représente le nom de l'unité et le nom du fichier à charger. Le chargement se fait sous le nom de l'unité sous lequel le programme a été sauvegardé.

L'instruction LOAD chargera toutes les variables et le programme BASIC qui se trouvaient en mémoire avant le chargement du programme spécifié.

L'instruction LOAD va également charger tous les fichiers liés au programme en question. Cependant, si l'option A est utilisée, le programme chargé sera dans son état de chargement et tous les fichiers créés avant seront chargés avec.

Ces lors, LOAD avec l'option (,A) peut être utilisé pour charger plusieurs programmes. Des instructions pourront être passées à ces programmes à l'unité.

Si l'option (,A) n'est pas programmée, le système charge le programme et revient à l'instruction OK du BASIC. Vous pourrez alors lister le programme.

LOAD"PROGRAM2.BAS" charge le programme PROGRAM2.BAS à partir du fichier correspondant sélectionné et revient à l'instruction OK du BASIC.

LOAD"JEU.001",A charge le programme JEU.001 à partir du lecteur A; et l'exécute à l'adresse de ce programme immédiatement.

IO P0="JEU.001" On peut également extraire le nom du programme dans une variable.

Les deux programmes suivants illustrent le passage d'une variable d'un programme à un autre par un fichier ouvert dans le premier programme et qui se sera fermé lors du chargement du second programme par l'instruction LOAD avec l'option (,A). Le début des instructions 20-30-50-60 du premier programme et des instructions 20-30 du second programme vous sera donné plus loin.

```
10 CLS
20 OPEN "VARIABLE" AS #1 :
30 FIELD #1, 255 AS Z0 :
40 INPUT "DON NOM ?" :
50 LET Z0 = A0
60 PUT #1,1
70 LOAD "PROGRAM2.BAS",A
```

1 PRENRE PROGRAMME
2 Ce programme demande votre nom,
3 le sauve dans un fichier et
4 appelle le programme PROGRAM2.BAS
5 sans relancer le lecteur VARIABLE

```

10 CLS
20 FIELD #1, 255 AS T$ : DEUXIEME PROGRAMME
30 GET #1, I : Rpprll par le 1er programme et
40 PRINT "TON NOM EST 'T$' : I : si premier programme.

```

3.1.3 BSAVE

BSAVE <spéc. fichier> , <départ> , <fin> <exécution>

Sauve un programme en langage BASIC résident en mémoire de l'adresse <départ> (initialement 0) jusqu'à l'adresse <fin> (initialement 255) sur le fichier pour le nom donné par <spéc. fichier>.

Si le nom du fichier n'est pas précisé dans <spéc. fichier>, le programme sera sauve sur le lecteur actuellement sélectionné. L'adresse d'exécution optionnelle permet de préciser à quelle adresse l'exécution du programme devra commencer lors d'un réajustement futur de ce programme. En son absence, le programme commencera à l'adresse de <départ>.

BSAVE <spéc. fichier> , <départ> , <fin> , 0

Cette dernière formulation est identique à la première, sauf que le code résident sur disque n'est pas un programme en langage BASIC résident en mémoire RAM, mais plutôt une image stockée dans le mémoire RAM du Vidéo Processor et puis dès lors l'adresse minimale est 0000H et l'adresse maximale est 3FFFH pour un MSX1 et 7FFFH pour un MSX2.

```

10 KEY 1, "PRINT"
20 KEY 2, "INPUT"
30 KEY 3, "UDOT"
40 KEY 4, "LOCATE"
50 KEY 0, "DATA"
60 KEY 6, "SCREEN"
70 KEY 7, "BLOAD"
80 KEY 9, "CIRCLE"
90 KEY 0, "LEFT"
100 KEY 10, "RIGHT"
110 BSAVE "FUNCTION.KEY", 0, 255

```

Sauve les positions RAM 0 à 255 sur le lecteur courant pour le nom "FUNCTION.KEY". Cette zone de mémoire contient les touches assignées aux 10 touches de fonction. Ainsi, si vous modifiez le contenu du texte des touches de fonction, il vous faudra de sauvegarder ce fichier pour le rétablir comme à l'origine.

```

10 SCREEN 2
20 CIRCLE (128, 96), 95, 15, 1, 1
30 A$ = "CIRCLE.PIC"
40 BSAVE A$, 0, 255, 0, 255
50 SCREEN 0
60 PRINT "DESSIN SAUVE SOUS LE NOM : "A$
70 END

```

Le programme dessine un cercle blanc et sauve l'image dans un fichier appelé "CIRCLE.PIC". Contrôlez puis le nom du programme peut aussi être une variable.

```

10 FOR I=0 TO 255
20 READ A$
30 POKE I, VAL("&H"&A$)
40 NEXT I
50 BSAVE "RAPIDE.BIN", 0, 255, 0, 255
60 END
70 DATA 05, 21, 00, 00, 01, 00, 03, 78, 00, 56
80 DATA 00, 91, 21, 00, 10, 00, 9C, 00, 00, 79
90 DATA B4, 20, F8, 1C, 20, 60, 1E, 20, 10, E2

```

Ce petit programme enregistre un programme en langage BASIC à l'adresse 0000H. Il se sauve sur le lecteur par défaut sous le nom "RAPIDE.BIN" en précisant qu'il s'agit de l'adresse 0000H à l'adresse 0000H et sur son exécution doit commencer à l'adresse 0000H. Vous pouvez voir l'effet spectaculaire de ce programme en appuyant sur les touches de l'instruction BLOAD.

Chaque paramètre de l'instruction BSAVE peut être une dans une variable excepté l'option "S".

```

10 P$="RAPIDE.BIN"
20 BLOAD 0000 : E=0000 : J=0000
30 BSAVE S$, B$, J

```

3.1.4 BLOAD

BLOAD <spéc. fichier> [,R] [,dérivée]

Cette instruction charge le fichier précisé dans <spéc. fichier> en mémoire, et s'il n'est pas obligatoirement vu par le programme, il est sauvegardé par l'instruction BSAVE.

Si aucun dérivé n'est précisé, le programme contenu dans le fichier sera chargé à l'adresse précise au moment du chargement du fichier par BSAVE, sinon le dérivé sera ajouté à l'adresse.

L'option R permet de charger automatiquement l'exécution de ce programme à l'adresse précise au moment du BSAVE.

BLOAD <spéc. fichier> , 0 [,dérivée]

Cette dernière formulation sert à charger un fichier contenu dans une image dans le Vidéo Processor. L'image s'ajoute à l'adresse à partir de laquelle il n'y a plus de place pour un éventuel décalage.

BLOAD "FUNCTION.KEY" Rappel du texte des touches de fonction tel qu'il avait été sauvegardé dans l'exemple de l'instruction BSAVE.

```

10 SCREEN 2
20 BLOAD "CIRCLE.PIC", 0
30 GOTO 30

```

Charge l'image contenue dans le fichier CIRCLE.PIC en RAM

Charge depuis le disque A: le fichier RAPIDE.BIN avec un décalage de 16 positions par rapport à l'adresse depuis laquelle il avait été sauve et lance l'exécution de ce programme à l'adresse précisée lors du BSAVE plus 16 positions. Ce programme affiche tous les caractères du MSX dans toutes les positions de l'écran. On peut l'arrêter en appuyant sur n'importe quelle touche.

```
10 SCREEN 0
20 INPUT "DONNEZ LE NOM DU DESSIN A AFFICHER":DS
30 SCREEN 2
40 BLOAD DS,S
50 GOTO 50
```

Le nom du programme à charger et le décalage peuvent être mis dans une variable. Cela permet, comme dans l'exemple ci-dessus, de choisir à partir du clavier quel fichier doit être chargé et à quelle adresse.

3.1.3 MERGE

MERGE <spéc. fichier>

C'est une commande et non une instruction. Cela signifie que, après son exécution, il provoque un retour à l'indicateur OK du BASIC, même s'il est inclus dans un programme.

MERGE permet de fusionner le programme BASIC ramené en mémoire avec un autre programme BASIC résident sur le disque dans le fichier précisé par (spéc. fichier). Le fichier (spéc. fichier) doit avoir été sauve en ASCII.

Fusionner signifie que tout le programme BASIC chargé se retrouvera en mémoire avec toutes les lignes du programme qui résidait en mémoire. Cependant, quand des lignes ont été le même numéro dans les deux programmes, celles qui résidaient en mémoire seront effacées au profit de celles en provenance du fichier.

Exemple: Le programme ci-dessous est sauve en ASCII sous le nom "FUSION.ASC" par l'instruction BSAVE "FUSION.ASC",A

```
25 PRINT "ET LEURS DISQUES"
35 PRINT "DESTINES A L'USAGE FAMILIAL"
40 END
```

Enfin maintenant le programme suivant après avoir pris soin de taper NEW pour effacer la mémoire de l'ordinateur.

```
10 CLR
20 PRINT "LES ORDINATEURS MSX"
```

```
30 PRINT "SONT DE BEAUX APPAREILS"
35 END
```

Ce petit programme fonctionne et qu'il est encore une fois allumé le fusionner avec le programme FUSION.ASC en posant:

MERGE "FUSION.ASC"

Un LIST nous fera découvrir le programme résultant:

```
10 CLR
20 PRINT "LES ORDINATEURS MSX"
25 PRINT "ET LEURS DISQUES"
30 PRINT "SONT DE BEAUX APPAREILS"
35 PRINT "DESTINES A L'USAGE FAMILIAL"
40 END
```

On voit que les lignes 25 et 40 ont été effacées et que la ligne 35 a été remplacée par celle du programme FUSION.ASC.

3.1.4 NAME

NAME <enc. minnnnnnn.see> AS <nouv. minnnnnnn.see>

Cette instruction change le nom d'un fichier en un nouveau nom. Le <enc. minnnnnnn.see> peut comporter un nom de unité de disques et doit comporter un nom de fichier qui existe déjà sur le lecteur.

Le <nouv. minnnnnnn.see> ne peut pas comporter de nom de lecteur autre que celui de <enc. minnnnnnn.see> et doit comporter un nom de fichier qui n'existe pas encore sur le lecteur.

Cette instruction ne s'applique pas le fichier; elle ne fait qu'en changer son nom. Si le nouveau nom existe déjà sur le disque, le message "File already exists" (fichier existe déjà) sera affiché.

Si vous précisez un nom d'unité différent dans <nouv. minnnnnnn.see> par rapport à <enc. minnnnnnn.see>, le message "Rename across disks" (changement de nom à travers deux disques) sera affiché.

Si le fichier <enc. minnnnnnn.see> n'existe pas sur le disque, le message "File not found" (fichier non trouvé) sera affiché. Dans aucun des trois cas d'erreur ci-dessus, le fichier ne changera de nom.

On peut utiliser les caractères de substitution et ? avec cette instruction dans le nom ou le résultat reste cohérent.

NAME "RAPIDE.BIN" AS "CAR-SHOW.BIN"

Rabaptées les fichiers RAPIDE.BIN en CAR-SHOW.BIN

NAME "0.BIN" AS "0.BIN"

Tous les fichiers se terminent par .BIN pour être les mêmes

nom que précédemment mais avec l'extension ".DSJ"

NAME 'JEU7.DSJ' AS 'GAME7.DSJ'

Cette commande se fonctionne pas ==> elle est incohérente. En effet, si le fichier JEU7.DSJ est trouvé, il va lire respectivement GAME.DSJ et son jeu GAME.DSJ car le '?' s'est pas P la même place dans les deux noms. Or, lors, si un fichier JEU7.DSJ est trouvé, il devrait aussi lire respectivement GAME.DSJ et son jeu à la place de JEU7.DSJ. La commande suivante aurait dû être : NAME 'JEU7.DSJ' AS 'GAME7.DSJ' ou la '?' est exactement la même place dans chaque nom.

Les deux noms de fichiers peuvent aussi être placés dans des variables et l'instruction peut être placée dans un programme comme dans l'exemple suivant :

```
100 P = '1.TXT' : N = '2.TE1'
110 NAME P AS N
```

Il est intéressant de remarquer les fichiers périphériques spéciaux "CON:", "AUX:", "LPT:", "PRN:" ou "NUL:"

3.1.7 KILL

KILL <spécification de fichier>

Cette commande détruit le fichier du lecteur. Cela signifie que le nom du fichier est retiré du Directory, le contenu du fichier est lui-même n'est pas effacé, mais de toute façon deviendra inaccessible. KILL peut être utilisé pour effacer quel type de fichier (programme, séquentiel, direct ou MIX DOS)

On peut également employer les caractères de substitution ! et ? dans le nom du fichier pour effacer plusieurs fichiers.

KILL 'ESB*.!B' efface le fichier ESB*.!B du lecteur.

KILL '!.!B' efface tous les fichiers ayant l'extension .!B du lecteur

KILL 'B:JEU,00?' efface du lecteur B: tous les fichiers ayant pour nom "JEU" et pour extension "00?". Le '?' va servir à effacer quel caractère, les fichiers JEU.001 - JEU.002 - JEU.00A seront effacés et, plus généralement, les fichiers se terminant par "JEU.00".

KILL '!.!' cette commande est valable mais il ne faut l'utiliser qu'avec précaution puisqu'elle détruit tous les fichiers de votre disquette.

```
10 INPUT "ENTREZ LE NOM DU PROGRAMME SANS L'EXTENSION : "
20 IF LEN(N) > 0 THEN PRINT "MAXIMUM 8 CARACTÈRES" : GOTO 10
30 IF INSTR(1, "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789") < 1 THEN PRINT "LE NOM SEUL S'APPLIQUE" : GOTO 10
40 N = N + ".BAS"
50 KILL N
60 END
```

Ce petit programme demande le nom du programme P, vérifie qu'il ne dépasse pas 8 caractères et qu'il n'a pas d'extension, sinon il affiche un message approprié. Il ajoute au nom du programme l'extension ".BAS" et efface enfin le programme.

3.1.8 FILES - LFILES

FILES <spec de fichier>

LFILES <spec de fichier>

Affiche P l'écran (FILES) ou sur l'imprimante (LFILES) le nom de tous les fichiers du lecteur courant.

Le paramètre <spec de fichier> peut être soit en son entier, soit un nom de fichier seul, soit une liste de caractères de substitution. Le paramètre P est employé dans le nom de fichier. Dans ce cas, les commandes FILES et LFILES vont afficher le ou les noms de fichiers trouvés par le disque et par correspondance P <spec de fichier>.

FILES Affiche P l'écran le nom de tous les fichiers présents sur le lecteur courant.

FILES 'B:' Affiche à l'écran le nom de tous les fichiers présents sur le lecteur B:

FILES 'JEU,00?' Affiche JEU,001 et ce fichier existe sur le lecteur courant ou FILE NOT FOUND s'il n'existe pas.

FILES 'A:RAPIDE.BIN' Affiche à l'écran RAPIDE.BIN et ce fichier existe sur le disque A: ou FILE NOT FOUND s'il n'existe pas.

FILES '!.BAS' Affiche P l'écran le nom de tous les fichiers présents sur le lecteur courant.

FILES 'B:BUDGET.!' Affiche P l'écran le nom de tous les fichiers du lecteur B: portant la extension BUDGET. Les fichiers BUDGET.1AN - BUDGET.REV - BUDGET.MAR - BUDGET.MAI

FILES 'BUDGET.*' Affiche P l'écran le nom de tous les fichiers du lecteur courant portant la extension BUDGET. Les fichiers BUDGET.1AN - BUDGET.REV - BUDGET.MAR - BUDGET.MAI

LFILES 'B:' Imprime sur papier le nom de tous les fichiers du lecteur B:

Voici un programme "MENU" simplifié qui se convient pour les programmes BASIC.

```

10 CLG
20 FILES
30 INPUT "QUEL PROGRAMME VOULEZ-VOUS?";P
40 LOAD P;R

```

3.1.9 COPY

COPY <epc de tichier source> TO <epc de tichier dest>

Cette instruction permet de copier un ou plusieurs fichiers d'un disque vers un autre ou vers le même. Vous pouvez aussi donner un nom de fichier de destination différent.

Le fichier <epc de tichier source> sera copié vers l'unité de fichier <spéc. de fichier dest>. Le paramètre <spéc. tichier dest> peut prendre trois formes :

1) S'il est composé du nom de l'unité de disquette seul, le fichier source sera copié sous le même nom vers l'unité de destination.

2) S'il est composé d'un nom de fichier seul, le fichier source sera copié sous le nom désigné sur l'unité courante.

3) S'il est composé d'un nom d'unité de disquette et d'un nom de fichier, le fichier source sera copié vers l'unité désignée, sous le nom désigné.

Si le fichier de destination existe déjà sur l'unité de destination, il sera remplacé par le copia du fichier source.

Reprenez-vous, si vous avez qu'un seul lecteur, le système va le partager en deux unités logiques A et B. Vous pouvez donc copier de disquette à disquette, la tâtâche vous permettrait de lire le fichier A sur la disquette A, tantôt la disquette B dans votre unique lecteur (voir chapitre 2.1).

COPY "RAPIDE.BIN" TO "B";

Le fichier RAPIDE.BIN de l'unité A sera copié vers le lecteur B sous le même nom.

COPY "RAPIDE.BIN" TO "VITE.BIN"

Le fichier RAPIDE.BIN du lecteur courant sera copié sur le même lecteur sous le nom VITE.BIN. Il y a donc maintenant deux fichiers identiques sur le même disque, mais sous des noms différents.

COPY "A:RAPIDE.BIN" TO "B:VITE.BIN"

Le fichier RAPIDE.BIN de l'unité A sera copié sur le lecteur B sous le nom VITE.BIN.

On peut, bien entendu, employer des caractères de substitution dans les noms de fichiers pour autant que

le résultat reste cohérent.

COPY "A:1.BAS" TO "B:"

Copie tous les fichiers de la disquette A ayant l'extension .BAS vers le lecteur B sous le même nom.

COPY "A:1.BAS" TO "B:1."

Copie tous les fichiers de la disquette A ayant l'extension .BAS vers le lecteur B sous le même nom mais avec extension.

COPY "B:BUDGET.78?" TO "A:COMPTE.s"

Copie tous les fichiers du lecteur B qui s'appellent BUDGET et dont l'extension contient un "A" se trouvant à la position de l'extension, vers le lecteur A sous le nom COMPTE avec la même extension.

On peut aussi placer l'instruction COPY dans le programme et placer ses paramètres dans des variables.

```

10 INPUT "NOM DU PROGRAMME A COPIER";ST
20 INPUT "NOM DU PROGRAMME COPIE";CT
30 COPY ST TO CT
40 END

```

Il est aussi intéressant de noter que l'on peut employer les fichiers techniques suivants : CON, LSI, PRN, NUL, AUX (voir chapitre 2.1).

```

COPY "CON" TO "A:ESSAI.TXT"
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Après avoir entré la commande COPY, faites RETURN et tapez votre texte en appuyant sur RETURN chaque fois que vous devez passer à la ligne. Pour provoquer le survol de votre texte dans le fichier ESSAI.TXT vous devez appuyer sur la touche F2 au début d'une ligne et puis faire RETURN. Après le survol vous reviendrez à l'invite OK du BASIC.

COPY "A:ESSAI.TXT" TO "CON"

Cette commande fera apparaître à l'écran le texte précédemment écrit sur le disque. Il faut bien entendu que le fichier ESSAI.TXT soit en ASCII.

COPY "A:ESSAI.TXT" TO "LST"

Provoque l'impression sur papier du fichier ESSAI.TXT.

3.1.10 RUN

RUN <spec de fichier> (,R)

Cette instruction avertit le programme BASIC spécifié par <spec de fichier> en mémoire et l'exécute immédiatement. L'instruction RUN détruit toutes les variables et toutes les lignes du programme qui aurait été présent en mémoire avant la RUN et fermera tous les fichiers laissés ouverts par ce programme. Si l'option (,R) est employée alors les fichiers ouverts par le programme précédent ne seront pas refermés.

RUN "JEU.BAS" Charge le fichier JEU.BAS et lance l'exécution du programme.

On peut mettre l'instruction RUN dans un programme et la paramétrer <spec de fichier> dans une variable.

```
110 P$="JFU.BAS"
120 RUN P$
```

L'option (,R), talant les fichiers du programme précédent ouverts, autorise le chaînage de plusieurs programmes si permet de passer des variables de l'un à l'autre via un fichier.

```
10 CLS
20 INPUT "TON NOM:"N$
30 INPUT "TON AGE:"A
40 OPEN "VARIABLE" AS V:
50 FIELD #1, 248 AS Z10, R AS Z24
60 LET Z10 = N$
70 LET Z24 = MND$JA1
80 PUT #1,1
90 RUN "PROGRAM.2",P
```

PROGRAMME "PROGRAM.2"

```
10 CLS
20 FIELD #1,248 AS Z10, S AS Z24
30 GET #1,1
40 A = CVD(Z24)
50 PRINT A;"EST L'AGE DE Mr "Z10
60 KILL "VARIABLE"
70 END
```

On peut également employer les noms de périphériques spéciaux COM et AUX. Seul AUX présente de l'intérêt lors d'une connexion entre deux ordinateurs par interface RS232.

3.1.11 MAXFILES

MAXFILES = <expression>

Permet de spécifier le nombre maximum de fichiers qui pourront être manipulés simultanément ou si vous préférez qui pourront être manipulés concurrentement.

Si <expression> vaut 0, alors seules les instructions SAVE - LOAD, BSAVE - BLOAD et MERGE pourront fonctionner mais aucun fichier ne pourra être ouvert (voir OPEN).

Par défaut, le système fixe MAXFILES à 1 et autorise donc la manipulation d'un seul fichier.

Si MAXFILES est fixé à une valeur supérieure à 4, les six premiers fichiers pourront être des fichiers disque. Cela revient à dire qu'au niveau des disques, le nombre de fichiers manipulés simultanément ne pourra jamais dépasser 6 même si MAXFILES est fixé à 15.

Si vous souhaitez utiliser 6 fichiers disque avec un fichier imprimant (LPT1), un fichier écran texte (CPT1), un fichier écran graphique (GPT1) et un fichier cassette (CAS1), alors programmez un MAXFILES=10 et réservez les fichiers 1 à 4 pour les fichiers disque et les numéros 5 à 10 pour les autres périphériques.

L'instruction MAXFILES réserve de l'espace mémoire sous la forme de communication des disques à raison de 267 octets par fichier (voir chapitre 3). Ainsi, un MAXFILES=15 va diminuer le nombre d'octets libres de 3738 par rapport à la valeur par défaut.

Attention!! Mettez toujours l'instruction MAXFILES au début de votre programme et avant toute déclaration de variables. Ne mettez pas non plus MAXFILES dans une sous routine qu'on accède par GOSUB. En effet, la modification de la mémoire réservée provoque l'effacement des variables et de la pile (STACK). Cette remarque vaut aussi pour l'instruction CLEAR. Je la mentionne ici car aucune documentation de fabricant n'en parle.

```
10 CLEAR 1000,WHDFE
20 MAXFILES=15
30 ...
```

CALL est une instruction de base normal qui provoque le exécution de l'instruction dont le nom suit immédiatement le mot CALL dans des Rôles d'instruction de base. Le contrôleur disque contient une ROM qui comprend deux instructions exécutées.

3.2.1 CALL FORMAT

Cette commande permet de lancer une disquette vierge ou d'ailleurs complètement une disquette qui contient déjà des fichiers.

Dès que vous aurez saisi la commande et on lance RETURN, l'ordinateur affichera :

Drive disk ? (e,é,...)

pour vous demander dans quel lecteur vous souhaitez insérer la disquette à lancer. Ensuite, il demandera du fabricant de votre ordinateur disque, un message de choix de format disque qui lui sera proposé. Par exemple :

- 1 - Single side
- 2 - Double side

répondre y de sélectionner la documentation concernant votre lecteur de disquette. Le message :

Strike a key when ready

sera affiché, ce qui signifie "Appuyez une touche de votre clavier". Ensuite, l'ordinateur le disquette et formatera même le lecteur choisi et lancera une disquette quelconque. Le processus de la disquette commence immédiatement et le message :

Format complet

sera affiché dès que l'opération sera terminée. Notez également que l'instruction CALL FORMAT peut être abrégée en _FORMAT.

3.2.2 CALL SYSTEM

CALL SYSTEM permet de quitter le BASIC pour retourner au MSX DOS. Cette commande n'est valide que si le Disk-diskette a été marqué à partir du MSX DOS/veroyes, et ce aussi, le chapitre 3, position RAM MSX64.

Pour cette commande, laissez le clavier ouvert en Basic et appuyez sur la touche de la commande en appuyant sur la touche. CALL SYSTEM peut être abrégé en _SYSTEM.

3.3 La manipulation d'un fichier séquentiel

Jusqu'à présent, nous avons vu les commandes de gestion des fichiers et comment ouvrir et fermer un programme ou d'une disquette, mais le Disk-diskette permet aussi de créer, de lire ou d'écrire des fichiers qui se manipulent par le programme mais aussi par les données.

On pourrait, par exemple, dans un programme qui permet de consulter un fichier bibliographique, de lui ajouter ou enlever des lignes ou même d'en modifier.

Un fichier séquentiel est celui où toutes les informations qu'on y dépose sont mises bout à bout avec simplement une marque de séparation entre chaque donnée. Le langage de programmation peut varier de tout normal, il est facile à employer mais a comme inconvénient de devoir lire tout ou partie en séquence.

Supposons que nous désirons créer un fichier d'adresses de nos parents et connaissances. Dans ce cas, le type séquentiel convient le mieux, car pour créer le fichier, il faudra introduire le nom des personnes par ordre alphabétique. Ce qui n'est pas facile et précis, lors de la constitution de ce fichier, il faudra lire toutes les informations précédentes celles de la personne sélectionnée avant de pouvoir renvoyer son adresse.

L'emploi idéal d'un fichier séquentiel est celui où les informations à saisir dans un fichier sont connues dans un ordre chronologique et où le langage de programmation ne peut pas lire et écrire.

À titre d'exemple, le journal d'un journal personnel. Ce jour, dans un journal les informations sont notées jour après jour et le contenu de ces informations peut varier énormément quant à leur aspect et à la longueur de chacune d'entre elles. D'autre part, le lecteur d'un journal s'intéresse à une information qui se trouve en séquence. Comment connaître l'évolution de sa santé ou l'état d'esprit de quelqu'un en lisant ce qu'il a écrit dans son journal le 3 mai que le 8 novembre et enfin le 23 février ?

Nous avons bien dit dans le paragraphe précédent qu'il s'agit d'un emploi idéal du fichier séquentiel, heureusement il existe beaucoup d'autres techniques. Alors, quand le langage d'accès à une information est le même que un élément critique pour le choix du type de fichier, alors le fichier séquentiel est certainement le meilleur grâce à sa facilité d'emploi.

Le fichier sur disquette, en général, se manipule facilement car il est facile d'adresser que nous avons vu dans le chapitre 3. Lorsque nous voulons connaître l'adresse d'une personne, nous appuyons sur ENTER le nom de la personne et nous obtenons son adresse. Ce qui est très facile et rapide. Ensuite nous appuyons sur ENTER, nous obtenons l'adresse qui nous intéresse ou nous appuyons sur ENTER une nouvelle adresse et

elle nous REPERFORME le format d'adresse.

De plus, les étiquettes en disquette, qu'elle soient séquentielles ou à accès direct, seront manipulées suivant une seule étape.

FONCTION LOGIQUE

INSTRUCTION BASIC

- | | |
|------------------------|----------------|
| 1- OUVERTURE | OPEN |
| 2- LECTURE
ÉCRITURE | INPUT
PRINT |
| 3- FERMETURE | CLOSE |

INPUT permet d'introduire dans une variable une donnée en provenance du lecteur et PRINT de permettre d'envoyer une information à une variable vers le lecteur.

3.3.1 L'ouverture d'un fichier séquentiel

OPEN <spin, fichier> FOR <accès> AS <numéro fichier>

L'instruction OPEN permet d'ouvrir le fichier désigné par <spin, fichier> sous une forme donnée décrite ci-dessous et de lier que le fichier sera l'information à lire ou à écrire par son <numéro fichier> global puis par son nom.

Il y a trois modes d'ouverture d'un fichier séquentiel :

FOR INPUT : signifie que le fichier est ouvert en lecture seulement. Commencez une ligne qu'il est interdit d'écrire de nouvelles informations ou de modifier des informations déjà existantes dans le fichier avec le mode d'ouverture, d'écrit ou, le fichier lui-même doit être ouvert en lecture ou en écriture.

FOR OUTPUT : Ce mode permet de lier et d'ouvrir le fichier en lecture seulement. On pourra de la sorte activer des informations à partir du début du fichier. Si le fichier n'existe pas sur le disque au moment de l'ouverture, il sera créé et sera en lecture seule. Si le fichier existe déjà au moment de l'ouverture, il sera d'écrit et sera en lecture seule. Dans ce cas, les données seront effacées.

FOR APPEND : Ce mode permet d'ouvrir le fichier en lecture et écriture. Toute information lue sera ajoutée au fichier. Les données déjà existantes dans le fichier ne seront pas effacées. Ce fichier doit donc être ouvert en lecture ou en écriture avant l'ouverture.

Le caractère <numéro fichier> doit être un numéro sous une forme à cinq chiffres, de 1 à 99999. Les numéros à moins de 15 chiffres sont complétés par des zéros à gauche.

pour autant dépasser le valeur lue par l'instruction MAXFILES. Ce numéro a une association au fichier dans le but que les autres instructions manipulant des fichiers puissent le reconnaître par son numéro plutôt que par son nom.

Cette association entre le nom du fichier et <numéro fichier> dure aussi longtemps que le fichier n'est pas fermé.

Pour techniquement l'instruction OPEN se divise en deux étapes : d'abord, les instructions d'ouverture et de fermeture du fichier, puis les instructions de lecture et d'écriture. Les données par le fichier sont lues ou écrites à l'aide de la commande OPEN. Les données lues ou écrites sont les données du fichier. Le numéro du fichier est le numéro du fichier. Le numéro du fichier est le numéro du fichier. Le numéro du fichier est le numéro du fichier.

En plus de cela, OPEN se termine par le fichier dans le fichier du disque, éventuellement par le fichier. Il va ensuite se que l'on appelle un FCB (File Control Block) le FCB sera écrit et enregistré dans le chapitre 11.

Un fichier séquentiel peut être ouvert en INPUT sous plusieurs numéros. Les données des autres modes lue ou écrites lue ou écrites par son nom de fichier. Les données lue ou écrites de cette instruction sont les données de 3.3.2.

3.3.2 L'écriture dans un fichier séquentiel

PRINT <numéro fichier> <liste d'expressions>

PRINT fonctionne exactement comme l'instruction PRINT bien connue et la différence est que les données sont écrites dans un fichier plutôt que sur l'écran. Il est interdit de lire ou d'écrire des données dans un fichier ouvert en lecture seule. Les données lue ou écrites sont les données de 3.3.2.

PRINT "1", "FAIT", "CHAUDE" montrera 1 FAIT CHAUDE sans séparation entre les trois mots. Il se sera de plus pour le PRINT sur un fichier, le but de l'écriture est de lier les données dans un fichier sous une forme lue ou écrite. Les données lue ou écrites sont les données de 3.3.2.

Le numéro du fichier est une constante, une variable ou une expression numérique qui spécifie quel fichier sera écrit. C'est l'instruction OPEN qui a associé le numéro au nom du fichier. Il est interdit de lire ou d'écrire des données dans un fichier ouvert en lecture seule. Les données lue ou écrites sont les données de 3.3.2.

La liste d'expressions est une liste de constantes, de variables ou de leurs expressions de type numérique ou chaîne de caractères séparées par les délimiteurs habituels de l'instruction PRINT, dans la liste '1' ou l'espace avant le juxtaposition des données et le code '1' l'écriture des données au début de la liste d'expressions suivantes. Si la

Ille des données ne se termine pas par '1' ou '1' la paire de code CR-LF (Carriage Return - Line Feed) sera écrite dans le fichier.

Les expressions numériques sont écrites dans le fichier telles qu'elles apparaissent à l'écran. Cela signifie que le nombre est précédé d'un espace s'il est positif ou d'un '-' s'il est négatif. De plus, chaque nombre est écrit du code espère comme décimal.

Les expressions de type réelle de caractères n'ont pas de délimitations rationnelles. Pour pouvoir distinguer les différences échappées lors de la fermeture du fichier, il faudra aussi écrire des délimiteurs. Le code réservé pour servir de délimiteur est le caractère.

Pour expliquer clairement l'impact de fichier suivant le contenu des instructions PRINT, nous allons prendre quelques exemples:

```
100 PRINT#1,10;-20;30;2
```

Cette première instruction envoie vers le fichier ouvert sous le numéro 1 les valeurs numériques 10, -20 et 40. L'image de ces données sur le disque sera la suivante:

```
_10_-20_40_CR_LF
```

Le symbole _1_ représente le code espace, CR représente le code retour chariot (CR) et LF le code de la ligne (LF) - 101.

Chaque valeur numérique est suivie d'un espace qui sert de délimiteur de telle sorte que nous pourrions plus tard relire ces valeurs du fichier et les attribuer à des variables numériques par l'instruction suivante:

```
320 INPUT#1,A,B,C
```

A contiendra 10, B contiendra -20 et C, 40.

Voici un extrait de programme qui écrit un nom et un prénom dans un fichier:

```
100 N$="DURAND"  
110 P$="ALBERT"  
120 PRINT#1,N$;P$
```

L'image écrite dans le fichier sera exactement identique à ce qu'on PRINT vers l'écran sur un terminal à savoir:

```
DURANDALBERTCR_LF
```

Nous constatons qu'aucun séparateur n'a été placé entre DURAND et ALBERT. Si nous allons plus tard en fichier pour relire les valeurs les variables N\$ et P\$ par l'instruction:

```
140 INPUT#1,N$,P$  
150 PRINT#
```

Nous aurons probablement surpris de constater que se

contient DURANDALBERT; cela est dû au fait qu'aucun séparateur n'a été placé dans le fichier lors de PRINT. Nous devons donc nous rappeler d'installer un séparateur de l'écriture dans le fichier. Le seul code de séparateur valable est le virgule. Modifions donc notre programme comme suit:

```
100 N$="DURAND"  
110 P$="ALBERT"  
120 PRINT#1,N$,";"P$
```

Nous obtenons alors l'image suivante dans le fichier:

```
DURAND;ALBERTCR_LF
```

Et si nous relisons le fichier, celle fois READ sera affecté à N\$ et ALBERT à P\$.

Un dernier problème surgit maintenant. En réel, si on écrit chaîne de caractères tel "CITROEN, 2CV, 1984", avec voyez que des virgules ont parties de la chaîne en elle-même et dès lors, lors de la lecture, celle chaîne sera interprétée comme 3 références distinctes.

La parade à ce problème est d'encadrer le chaîne de guillemets dans le fichier. Mais comment faire pour que ces guillemets s'écrivent dans le fichier. Et bien, nous allons de cette façon qu'on PRINT écrit à l'écran, nous employerons le caractère CHR\$(34).

Nous aurons qu'il faut entourer le chaîne de guillemets lorsque celle-ci contient des espaces ou des caractères spéciaux les autres caractères, ou encore si elle contient des virgules, des points-virgules ou des codes CR-LF intégrés à la chaîne.

```
100 A$="CITROEN, 2CV, 1984"  
110 PRINT#1,CHR$(34);A$;CHR$(34)  
120 PRINT#1,CHR$(34);"OPEL, KADET, 1982;ENCHONNAGE";CHR$(34)
```

L'image suivante sera alors déposée dans le fichier:

```
"CITROEN, 2CV, 1984"OPEL, KADET, 1982;ENCHONNAGE"CR_LF
```

Lors de la lecture du fichier, la présence de guillemets autour des deux chaînes provoquera l'interprétation exacte des données.

```
190 INPUT#1,D16,D26 D16 = CITROEN, 2CV, 1982  
D26 = OPEL, KADET,  
1982;ENCHONNAGE
```

Exemples de syntaxe:

```
PRINT#2,"BONJOUR"  
PRINT#3,27;"42"  
PRINT#1,CHR$(34);ATTENTION";CHR$(13);CHR$(10);CHR$(13)  
PRINT#5,A$;B$;  
PRINT#4,A$;B$;C$;D$;E$;  
PRINT#6,A$;B$;"1";C$  
PRINT#4;2;A$;0
```

3.3.3 La lecture d'un fichier séquentiel

```
INPUT<Numero Linhas>, <lista de variáveis>
```

INPUT: lit en ou des lilles d'un fichier aléatoire et les
noms d'une ou des variables de données.

«**waitfor** **finisher**» est une instruction, une variable ou une expression numérique qui spécifie quel **finisher** sera le, C'est l'instruction **OPEN** qui a besoin du nombre non de **finisher** il faut donc introduire la valeur qu'on veut dans l'**OPEN** du **finisher** qu'on considère. Ce nombre on peut dire **finisher** qui sera toujours le nombre de l'instruction **WAITFOR** avec de cette façon on aura une l.

<item de mailbun> tape dans une ou plusieurs mailles
répondre par la mode virgin dont le type
l'écrit-Blepi-Boupi-Charcut doit correspondre aux l'écrit
les de linher avec gain d'ordr lygn mignith, il sera
en autant d'élements du fichier qu'il y a de mailles en
cette de variabls

Dăruirea este un dar de la Dumnezeu la oameni, dar
 Dumnezeu nu este un Dumnezeu care dăruiește
 darurile sale oamenilor, ci un Dumnezeu care
 dăruiește oamenilor darurile sale.

[illegible][illegible]

Il y a premier ierastien n'est qu'un guillemet, la chaîne
est considérée comme terminée de l'appellation d'une
ligature, d'un code CR, d'un code LE ou si 255 caractères ont
été utilisés dans la chaîne.

Il y a une marque de fin de fichier (EOF) qui indique la fin d'un fichier. Dans ce cas, la variable en cours de compilation sera traitée. Si cette variable n'est pas la décalée de <input> la variable de la message "input qui est un" input se décale de fin de fichier sera utilisée.

Exercice de synthèse

```
INPUT I, A
INPUT I, AM
INPUT I, C, DQ, NQ
INPUT I, S, A, AG, T, I, T, ID
INPUT I, B, I, C, DQ, EQ, F
INPUT AN=1, A
```

S.3.4 La signature d'un fichier aléatoire

CLOSE **IN** <numero lichter>1, **IN** <numero lichter>2, ..., **IN**

Elimination CLOSE (former) social 100 options
d'abolition/abolition des options sociales.

Siempre l'hiere en una sonata a-ritmo, una melodia
 musical que organiza musical que agita en quel
 l'hiere porta l'hiere, l'hiere OPEN que
 melodia un melodia en son de fustil. El fustil
 l'hiere en la melodia indica que l'hiere OPEN de l'hiere
 l'hiere en el melodia que l'hiere melodia agito. Co melodia
 melodia l'hiere l'hiere l'hiere melodia melodia
 l'hiere de MAXFIRE avec de tous l'hiere melodia de

Une limitation CLOSE peut limiter plusieurs lignes à la fois, indiquant simplement la position du fichier cible, après que son volume, si CLOSE ne contient aucun nombre de lignes, tous les fichiers cibles sont fermés.

L'Association en la hon de l'hislois el son univirs el
leratus s'es la lis de l'instruction Culiel, la licher goul
dis los illis licheris sur un papié grislonque. De même,
ce papié pouvoit être rempli pour ouvrir l'imporio qui
délivre.

L'installation CDOE est la plus importante pour les fichiers
monnaie au mode CHPTM ou APPHD, car c'est elle qui gère les
les modes de la de fichier LITH est ainsi dans la banque
monnaie de fichier et que la banque LITH est ainsi sur la
disque.

Compte, CLDIE au libellé d'indemnité de direction
correspondant à sa fonction n'y indique la somme
longueur du libellé, si le date et l'heure du NEX21 de la
indemnité de la fonction, La C&I sera aussi la liste sur la
disque.

Les institutions CND, CLEA et NCM de mise en œuvre de la médiation d'un processus prévoyant également la mise en œuvre de la loi de la médiation.

Exemple de système :

```

CLOSE
CLOSE@
CLOSEI
I CLOSEI 1, 12, 34
CLOSEI 3, 8
CLOSEA, T
CLOSEM, I

```

3.3.3 exemples de programmes avec fichier séquentiel

3.3.3.1 création ou extension d'un fichier d'adresses

```

10 CLEAR$OO:MAXFILES=1
20 ON ERROR GOTO 170
30 OPEN 'ADRESSE.TXT' FOR APPEND AS #1
40 GOTO 60
50 OPEN 'ADRESSE.TXT' FOR OUTPUT AS #1
60 CL0
70 INPUT 'NOM' $#
80 INPUT 'PRENOM' $#
90 INPUT 'RUE' $#
100 INPUT 'No' $#
110 INPUT 'CODE POSTAL' $#
120 INPUT 'LOCALITE' $#
130 PRINT #1, $#, $#, $#, $#, $#, $#
140 PRINT 'RETURN POUR CONTINUER ESC POUR ARRETER'
150 A$=INKEY$:IF A$=" " GOTO 150
160 IF A$="C" GOTO 130
170 IF A$="D" GOTO 140
180 CLOSE #1:CL0
190 IF A$="D" AND A$="D" THEN RESUME 30
200 IF A$="D" GOTO 220
210 PRINT 'Disque protégé contre l'écriture'
220 IF A$="D" GOTO 240
230 PRINT 'Ecr de disque dans le lecteur'
240 ON ERROR GOTO 0
250 INPUT 'Veuillez Retirer avec précaution le disque'
260 RESUME

```

Les lignes 10 et 20 servent à tester l'existence éventuelle d'un fichier, à l'absence de ce dernier, le programme crée le fichier. Les lignes 30 et 40 servent à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier.

Si le programme n'a été exécuté qu'une seule fois, les lignes 30 et 40 servent à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier. Les lignes 30 et 40 servent à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier.

Les lignes 60 et 120 permettent d'introduire les données relatives à une personne, la ligne 130 va sauvegarder les données des variables \$#, \$#, \$#, \$#, \$#, \$# sur le disque avec une séparation après chacune d'entre elles. La ligne 140 va sauvegarder le contenu de la variable \$# sur le disque avec une séparation après chacune d'entre elles. La ligne 150 va sauvegarder le contenu de la variable \$# sur le disque avec une séparation après chacune d'entre elles.

Les lignes 170 et 180 vont demander s'il y a encore des adresses à entrer, auquel cas on passe à la ligne 30, ou, si l'encodage est terminé, on passe à la ligne 190. Cette dernière ligne efface l'écran et affiche le programme.

Les lignes 200 et 240 créent une petite boucle de protection contre les erreurs de saisie. La ligne 200 affiche un message on est sur le disque protégé contre l'écriture et la ligne 210 est une instruction qui provoque le message de l'écriture qui a été l'écriture.

3.3.3.2 Consultation d'un fichier d'adresses

```

10 CLEAR$OO:MAXFILES=1
20 ON ERROR GOTO 170
30 OPEN 'ADRESSE.TXT' FOR INPUT AS #1
40 CL0
50 INPUT 'DONNEZ LE NOM RECHERCHE' $#
60 INPUT 'DONNEZ LE PRENOM' $#
70 INPUT $#, $#, $#, $#, $#, $#
80 IF A$="C" OR A$="D" GOTO 10
90 PRINT
100 PRINT $#, $#, $#, $#, $#, $#
110 PRINT 'NOM' $#, 'PRENOM' $#
120 PRINT $#, $#, $#, $#, $#, $#
130 PRINT $#, $#, $#, $#, $#, $#
140 INPUT 'AUTRE RECHERCHE O/N' $#
150 CL0
160 IF A$="C" GOTO 30
170 IF A$="D" AND A$="D" THEN PRINT 'Disque protégé contre l'écriture'
180 IF A$="D" THEN PRINT 'Ecr de disque dans le lecteur'
190 IF A$="D" THEN PRINT 'Disque protégé contre l'écriture'
200 ON ERROR GOTO 0
210 INPUT 'Veuillez Retirer avec précaution le disque'
220 RESUME

```

Les lignes 10 et 20 servent à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier. Les lignes 30 et 40 servent à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier.

La ligne 50 sert à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier. La ligne 60 sert à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier.

Quand la comparaison est terminée, les lignes 80 et 130 affichent les données du fichier pour cette personne. La ligne 140 sert à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier.

La ligne 150 sert à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier. La ligne 160 sert à tester l'existence d'un fichier, à l'absence de ce dernier, le programme crée le fichier.

l'ensemble Europe c'est le seul moyen pour pouvoir réaliser la dernière. Après le début, si le fichier n'est simplement jamais ouvert, une autre recherche sera possible qu'il est possible de l'ouvrir. Le fichier se trouve peut être avant.

La ligne 160 laisse votre réponse et provoque un saut à la ligne 30 si la réponse est "oui" ou termine le programme si la réponse est différente de "oui".

La ligne 170 est assignée au cas "Parfois". Si l'appel de l'écriture 55 (appelé cas "oui"), n'est pas la non-recherche n'a pas pu être trouvé avant la fin du fichier. Un message approprié est alors produit et le programme réécrit ce message. Les lignes 180 à 220 affichent un message s'il n'y a pas de message dans le fichier ou s'il n'y a pas de fichier. ADRESSE.TXT dans la Pratique.

3.3.4 LINE INPUT

LINE INPUT (numéro fichier), <variable> (type)

LI est utilisé pour ouvrir sous le <numéro fichier> une ligne complète sans autre compte des caractères conventionnels comme l'espace ou la virgule. Les données sont assignées à la variable <variable> suivant, CALLP et doit obligatoirement être de type string.

Le fichier peut avoir été ouvert en mode INPUT. Cette instruction lit tous les caractères y compris l'espace, la virgule et les caractères de fin de ligne jusqu'à ce qu'un code CR (carriage return) soit rencontré ou jusqu'à la fin du fichier soit combiné avec 255 caractères. La séquence CR-LF est considérée comme permettant ainsi au prochain LINE INPUT de lire les caractères de la ligne suivante de la même manière.

Cette instruction permet donc de visualiser le contenu d'un fichier aléatoire ou ASCII comme dans le programme qui suit d'exemple 3.3.4.2.

<numéro fichier> est une expression numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui associe un numéro au nom du fichier. Il faut donc reproduire la même instruction dans l'instruction OPEN du fichier sur lequel on veut lire la même instruction. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers ouverts par MAXFILES avec de toutes façons un maximum de 6.

```
10 CLRF 512 : MAXFILES = 1 : CLS
20 ON ERROR GOTO 60
30 OPEN "ADRESSE.TXT" FOR INPUT AS #1
40 LINE INPUT #1, #1
50 PRINT #1 : GOTO 40
60 IF ERR=55 THEN CLOSE#1 : END
70 ON ERROR GOTO 60
```

Ce programme affiche le contenu réel du fichier ADRESSE.TXT.

3.4 Les fonctions spécifiques aux fichiers séquentiels

3.4.1 INPUT

INPUT (<numéro fichier>), (<numéro fichier>)

Cette fonction permet de lire le fichier portant le <numéro fichier> un certain nombre de caractères, dans une position <car>, <position>, tous les caractères <y compris les caractères> sont passés au point de code <N>, puis au cas de la fin de fichier.

C'est donc la seule fonction ou instruction qui permet de lire le fichier de type Po Pointé qui se trouve dans un fichier séquentiel.

Exemple Programme utilisant la fonction POI P'un fichier séquentiel. Les codes CR et LF sont visualisés à l'écrit en mode string. La seule ligne de fichier sera affichée par POI du fichier. Cependant, il n'y a pas de CLOS. En effet, l'instruction END inclut un CLOS de tous les fichiers ouverts.

```
10 MAXFILES = 1
20 CLS
30 OPEN "ADRESSE.TXT" FOR INPUT AS #1
40 A$=INPUT#1, #1
50 IF A$=CHR(13) THEN PRINT "CR:"; GOTO 60
60 IF A$=CHR(10) THEN PRINT "LF:"; GOTO 40
70 IF A$=CHR(10) THEN PRINT "LF:"; GOTO 40
80 IF A$=CHR(10) THEN PRINT "LF:"; GOTO 40
90 IF A$=CHR(10) THEN PRINT "LF:"; GOTO 40
100 GOTO 60
```

3.4.2 End Of File

EOF (<numéro fichier>)

EOF est l'abréviation de End Of File (Fin de Fichier). Cette fonction retourne -1 lorsqu'il se fin d'un fichier séquentiel a été atteint. Utilisez cette fonction avant chaque INPUT afin de prévenir l'erreur "INPUT #NBI END".

<numéro fichier> est une expression numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui associe un numéro au nom du fichier. Il faut donc reproduire la même instruction dans l'instruction OPEN du fichier sur lequel on veut lire la même instruction. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers ouverts par MAXFILES avec de toutes façons un maximum de 6.

```
10 MAXFILES = 1
20 CLS;WIDTH 39
30 OPEN "ADRESSE.TXT" FOR INPUT AS #1
40 IF EOF(1) = 1 GOTO 80
50 AS=INPUT$(1,1)
60 PRINT RIGHT$("D" + HEX$(ASC(AS)),2); " "
70 GOTO 40
80 PRINT
90 END
```

L'instruction de la ligne 40 peut être abrégée en :

```
40 IF EOF(1) GOTO 80
```

3.4.3 LOCate

LOC(<numéro fichier>)

Cette fonction retourne la localisation du pointeur du fichier séquentiel. En d'autres mots, elle indique combien de blocs de 256 caractères ont déjà été lus de (ou écrit sur) le fichier <numéro fichier>.

Tous les décodeurs y compris les codes ER et LF sont compris de même, bien sûr, que vos données. Si aucun bloc n'a encore été lu, la fonction retourne la valeur 1 car, lors de l'OPEN d'un fichier séquentiel en INPUT, le tampon mémoire réservé à un bloc est déjà rempli avec les 256 premiers caractères lus de votre fichier. Par contre en écriture, la fonction retourne bien 0 si vous n'avez pas encore écrit 256 caractères dans le fichier.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier; il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

Exemple: ce programme va remplir 6 blocs de 256 caractères avec les données spécifiées dans le but de créer un petit fichier séquentiel d'essai.

```
10 MAXFILES = 1
20 A# = "FICHIER DE TEST"
30 AZ = 147
40 A1 = 123456
50 A# = 12345678901234
60 PRINT #1,A#1;"1"AX101;A#
70 IF LOC(1)<6 GOTO 60
80 CLOSE #1
90 END
```

3.4.4 LOF Length Of File

LOF(<numéro fichier>)

La fonction LOF retourne la longueur en octets du fichier ouvert sous le <numéro fichier>.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier; il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

Exemple: ce programme vous donne la longueur du fichier posé en réponse à la question du programme.

```
10 CLS;MAXFILES=1
20 INPUT "DONNEZ LE NOM D'UN FICHIER":IF#
30 OPEN F# FOR INPUT AS #1
40 PRINT
50 PRINT F#; " a une taille de";LOF(1);"octets."
60 PRINT:END
```


3.3. La manipulation d'un fichier à accès direct

Le fichier à accès direct est moins simple à mettre en œuvre que le fichier séquentiel, comme nous l'avons indiqué au chapitre 2.2.3. Contrairement au fichier séquentiel, les informations ne sont pas lues bout à bout mais organisées en enregistrements de longueur fixe et éparpillés.

La détermination de la longueur de l'enregistrement est liée à l'OPEN du fichier; elle ne pourra donc pas être changée durant la manipulation du fichier. De même, lorsque l'on ouvre un fichier à accès direct, on doit spécifier la longueur de l'enregistrement. On ne peut pas modifier la longueur de l'enregistrement pendant l'exécution.

L'inconvénient principal du fichier à accès direct réside dans le fait qu'il faut prévoir une longueur d'enregistrement suffisante pour que toute l'information qui doit être stockée dans l'enregistrement y trouve place. Or, il est parfois difficile de prévoir d'avance quelle grandeur sera la plus grande information. Il est possible aussi que des informations de petite taille soient stockées dans des enregistrements de grande taille, gaspillant ainsi de l'espace disque inutilement.

Par contre, l'absence d'attente liée dans le cas du fichier à accès direct est un avantage qui lui confère une rapidité de traitement qui n'est pas possible avec un fichier séquentiel. On n'a pas à parcourir l'ensemble du fichier pour trouver les données d'intérêt, on peut accéder directement à la donnée qui nous intéresse.

Un exemple concret d'application d'un fichier à accès direct pourrait être un fichier de stock de marchandises ou de pièces. Chaque pièce ou marchandise est stockée dans un emplacement unique. On pourrait alors servir de ce fichier comme un répertoire de stock. Chaque emplacement est identifié par un numéro unique. On peut alors accéder directement à la donnée qui nous intéresse, sans avoir à parcourir l'ensemble du fichier.

De plus, cette disposition, plus complexe qu'il se souvient de décrire, permet de stocker des informations qui changent fréquemment. C'est d'ailleurs ce qui est intéressant de la longueur de l'enregistrement, car on peut modifier la longueur de l'enregistrement sans avoir à modifier la longueur de l'enregistrement.

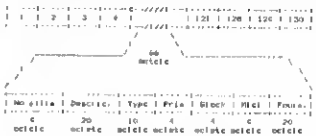
Une limitation spécifique du fichier à accès direct est la longueur de l'enregistrement. On ne peut pas avoir de fichiers à accès direct de longueur variable.

Autre part, dans le cas de données structurées, l'information est stockée dans l'enregistrement. Les variables numériques sont stockées dans des champs de données. Les variables de type caractère sont stockées dans des champs de données. Les variables de type caractère sont stockées dans des champs de données. Les variables de type caractère sont stockées dans des champs de données.

est elle, deux fonctions spéciales sont fournies par le BASIC qui permettent, l'une de disposer la variable d'accès à l'enregistrement et l'autre de calculer la valeur numérique de l'enregistrement. C'est la variable INCI/NOI et l'INCI/NOI.

C'est ainsi, l'écriture de l'INCI/NOI est une instruction obligatoire de l'écriture de la variable d'accès à l'enregistrement. L'écriture de l'INCI/NOI est une instruction obligatoire de l'écriture de la variable d'accès à l'enregistrement. L'écriture de l'INCI/NOI est une instruction obligatoire de l'écriture de la variable d'accès à l'enregistrement.

FICHIER DE 130 ENREGISTREMENTS



3.4. L'ouverture d'un fichier à accès direct

OPEN <type, fichier> AS <nom du fichier> LEN <longueur>

L'instruction OPEN ouvre le fichier à accès direct. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné.

Le OPEN ouvre un fichier à accès direct. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné.

Un fichier à accès direct peut être ouvert par plusieurs OPEN avec des noms différents. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné.

<type, fichier> est une chaîne de caractères qui indique le type de fichier et le nom du fichier. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné.

La variable 'numéro fichier' est une variable numérique qui indique le numéro du fichier. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné. Le fichier est ouvert en mode 'accès direct' et le nom du fichier est donné.

30 positions réservées pour la variable NO INE = NOM
 15 positions réservées pour la variable PE INE = PRENOM
 30 positions réservées pour la variable RS INE = RUE
 4 positions réservées pour la variable VO INE = NUMERO
 20 positions réservées pour la variable VI INE = VILLE
 --
 99 positions au total pour l'enregistrement.

Une instruction FIELD ne peut allouer un espace total plus grand que la longueur de l'enregistrement déclaré dans l'instruction OPEN. Dans notre exemple, le total de l'espace alloué est 99; donc, l'OPEN devra avoir été choisi avec une longueur d'enregistrement égale ou supérieure à 99. Sans quoi le message d'erreur 'Field overflow' sera produit indépendamment de l'apariel dans FIELD).

Attention!! L'instruction FIELD ne place jamais de zéros dans le tampon du fichier. Ce sera la tâche des instructions RESET, SET.

Il faut noter aussi que l'espace réservé pour stocker des valeurs numériques doit être de 2 octets pour les valeurs entières, de 4 octets pour les valeurs à virgule décimale et de 8 octets pour les valeurs doubles précises. Cet espace NE DOIT PAS être alloué à une variable NUMERIQUE sans au contraire à une variable CHARNE. Des fonctions spéciales sont fournies pour l'attribution d'une valeur numérique dans la variable chaîne correspondante INE1 MD5 MD6).

On peut décrire un état enregistré sous plusieurs formats différents par plusieurs instructions FIELD. Ces différents formats seront indiqués par l'ordinateur pour autant qu'aucune variable chaîne ne porte la même adresse mémoire et que l'unique découpage indique le lorsque complet de l'enregistrement.

Exemples:

```
10 OPEN 'BRIK55.13' AS 1 LEN = 54
20 FIELD 01,1 AS P4,11 AS N6,11 AS N6,11 AS AN,11 AS V6
30 FIELD 01,11 AS X6,22 AS N6,11 AS X6,11 AS X6
40 FIELD 01,13 AS X6,22 AS N6
```

P4 = PRENOM
 N6 = NOM
 N6 = NOM ET CONJOINT
 RUE = RUE ET No
 VI = VILLE
 N6 = NOM GLOBAL (i.e.d. NOM + NOM ET CONJOINT)
 X6 = ADRESSE COMPLETE (i.e.d. RUE + No + VILLE)
 X6 = Variable d'ID

Supposons que la façon habituelle d'enregistrement suit:

```
JEAN MARIE DUPOND      DURAND      DE PARIS, THORSETTE
-----
```

Dans ce cas, les variables suivantes assigneront:

Je = 'JEAN-MARIE'
 N6 = 'DUPOND'

La parcella <longueur> est une constante, une variable ou une expression numérique indiquant la longueur des enregistrements de ce fichier. Si la constante n'est pas égale, la valeur par défaut est 256 octets. La longueur minimale autorisée est 1.

Un fichier ouvert en mode "accès direct" peut être lu par l'instruction GET et écrit par l'instruction PUT tant qu'il reste ouvert.

Exemples de syntaxe

```
10 OPEN 'AISTOCK.DIR' AS 01 LEN=66
10 OPEN 'STOCK.DIR' AS 01 LEN=66
10 OPEN 'DIR.1.DIR' AS 1
```

```
10 AS 'AISTOCK.DIR' LEN=66
20 OPEN AS 01 LEN=66
30 OPEN AS 01 LEN=66
```

3.3.2 FIELD Création de champs formats

FIELD Définition du fichier, longueur, attributs, etc.

FIELD alloue une place et une longueur dans le tampon système réservé aux enregistrements de fichier et accès direct. Numéro attribué pour des variables de type chaîne de caractères. Avant qu'une instruction FIELD ou une écriture INE1 d'un enregistrement ne puisse être exécutée, il faut découper le tampon système du fichier avec l'instruction FIELD.

Chaque attribut est une constante numérique, une variable numérique ou une expression numérique qui apparaît au quel fichier porte l'instruction. C'est l'instruction OPEN qui a attribué au numéro du nom du fichier; il faut donc reproduire ce même numéro attribué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro se peut être attribué à la répétition au nombre de fichiers dirigé par NAINTELS avec les octets d'un ensemble de 6. Longueur est une constante, une variable ou une expression numérique déclarant la longueur de la zone réservée pour la variable chaîne dans le tampon système de l'enregistrement.

On peut répéter 'longueur' variables chaînes' autant de fois que le permet la longueur maximale de la ligne. Dans 1255 oct. est alloué de partiel l'enregistrement de plusieurs zones réservées indépendamment de chaque variable chaîne. Mais attention que toute instruction FIELD commence le découpage par le début de tampon système.

Exemples:

```
FIELD1, 30 AS N6, 15 AS P4, 30 AS N6, 4 AS N6, 20 AS V6
```

Cette instruction découpe le tampon système des enregistrements de fichier numéro 1 en:

```

N1 = 'DURAND'
N2 = 'DE DADID, 7'
V1 = 'PARSEILLE'
N3 = 'DUFOND' DUFOND
N4 = 'DE PADIE, ZYON'
N5 = 'JEAN-MARIE DUFOND' DUFOND

```

N'oubliez jamais une variable entrant dans la composition d'une instruction FIELD avec un INPUT ou un LEI ! Ici, vous ne pourriez plus obtenir les données correspondantes du langage de l'enregistrement.

EXEMPLE 1

```

10 OPEN 'ADRESSE.TXT' AS #1 LEN = 99
20 FIELD #1, 30 AS N1, 15 AS P1, 30 AS R1, 4 AS NR, 20 AS V1
30 FOR I=1 TO 10
40 GET #1, I
50 DDINT N1 : PRINT P1 : PRINT R1:NR : PRINT V1
60 NEXT I
70 END

```

Dans cet exemple 1, un seul découpage de l'enregistrement a été réalisé en ligne 20. Le boudé de 30 à 60 est r1 finit à l'arrêt le contenu de l'enregistrement 2 à 10 de finit 'ADRESSE.TXT'.

EXEMPLE 2

```

10 OPEN 'ADRESSE.TXT' AS #1 LEN = 99
20 FIELD #1, 2 AS NR, 97 AS N1
30 FIELD #1, 50 AS N1, 15 AS P1, 30 AS R1, 4 AS NR, 20 AS V1
40 GET #1, I
50 I=CVIN(I)
60 FOR I=2 TO 12
70 GET #1, I
80 DDINT N1 : PRINT P1 : PRINT R1:NR : PRINT V1
90 NEXT I
100 END

```

Dans cet exemple, deux découpages ont été réalisés le premier ligne 20 sert à donner le format de l'enregistrement dans lequel on a décidé de placer le numéro de dernier d'enregistrement du fichier variable NR1. Les 97 octets réservés à la variable N1 sont en sorte que l'écriture de la longueur de l'enregistrement sera réservée par cette instruction FIELD. Le deuxième découpage (ligne 30) sert à lier le découpage de tout les enregistrements à l'exception de premier.

Attention de bien veiller à l'emploi des variables ! En effet, lors de la lecture du record 1 (DET1,1), la variable N1 contiendra elle aussi les 15 premiers caractères du record 1 mais ne renverra pas correspondre pas à une donnée personnelle de la lecture des autres records. De même, NR1 ne contiendra le numéro du dernier enregistrement que lors de la lecture du premier enregistrement et non lors de la lecture des suivants.

EXEMPLE 3

Supposons que vous désiriez servir une table de 16 noms de

maxime 16 caractères chacun dans un enregistrement du fichier 'FICHIER.TAB'. Vous auriez de écrire le programme suivant :

```

10 DIM I$(15)
20 OPEN 'FICHIER.TAB' AS #1
30 FIELD #1, 16 AS I$(1), 16 AS I$(2), 16 AS I$(3), 16 AS I$(4), 16 AS I$(5), 16 AS I$(6), 16 AS I$(7), 16 AS I$(8), 16 AS I$(9), 16 AS I$(10), 16 AS I$(11), 16 AS I$(12), 16 AS I$(13), 16 AS I$(14), 16 AS I$(15)

```

Avant qu'il est tellement plus simple d'écrire le programme avec une petite boucle plutôt que de devoir taper cette très longue instruction comme vous le voyez l'exemple ci-dessous.

```

10 DIM I$(15)
20 OPEN 'FICHIER.TAB' AS #1
30 FOR N=0 TO 15
40 FIELD #1, N*16 AS N1$(N), 16 AS T$(N)
50 NEXT N

```

Remarquez alors que l'expression N*16 AS N1\$(N) ne sert à attribuer une longueur à une variable qui ne sert à rien (N1\$(N) sera qui correspondra à l'écriture de la variable T\$(N) par rapport au début de l'enregistrement. Rappelons aussi, en effet, que l'instruction FIELD recrée tous les jours le découpage au début du temps mémoire.

3.5.3 LSET (LEFT SET - PLACER À GAUCHE) / RSET (RIGHT SET - PLACER À DROITE)

```

LSET <X> = <Y>
RSET <X> = <Y>

```

Ces deux instructions servent à transférer le contenu de <Y> vers le variable <X> en alignant les données à gauche pour LSET ou à droite pour RSET. Habituellement, elles sont employées pour transférer <Y> vers le partie du temps d'un fichier à mots direct qu'une instruction FIELD préalable avait nommé <X>.

<Y> est une constante ou une variable, de type chaîne qui est l'information qui l'on veut placer dans le temps mémoire de l'enregistrement dans le but de l'écriture dans le fichier.

<X> est une variable de type chaîne qui doit avoir été définie par une instruction FIELD préalable.

Exemple :

```

10 OPEN 'ADRESSE.TXT' AS #1 LEN = 92
20 FIELD #1, 30 AS N1, 12 AS P1, 30 AS R1, 20 AS V1
30 LSET N1 = 'DURAND'
40 R1 = 'JEAN' : RSET D1 = P1$
50 R1 = 'AVENUE DE LA LIBERATION, 30'
60 LSET V1 = 'PARIS'
70 PUT #1, I

```

L'instruction FIELD dn in ligne 20 a prévu 30 caractères pour le nom (N1), 12 caractères pour le prénom (P1), 30 pour le rue (R) et finalement 20 pour le ville (V1). Cela signifie donc que dans l'instruction la longueur 30 caractères dans le tableau désigne le fichier ADRESSE.N1 pour le nom. Dans ce moment-là le nom à renvoyer dans le fichier est est plus petit ou égal grand que 30 caractères ?

11. Si la longueur de la chaîne est plus petite, la partie de la non-réserve dans le luxon mineur sera nommée avec des chiffres (code 32).

21 Si la longueur en inchure s'élève au plus grand, celle du pied sera l'équivalent au pied du 30ème caractère.

Experiments

Donn de 12 narvalières réservée à F. 1.

```

LSET P# = "JEAN"
RSET P# = "JEAN"
LSET P# = "NADUCHODNOZOR"
RSET P# = "NADUCHODNOZOR"

```

Les tirets repréhennt les espaces. Remarque : In
position de JEAN dans la zone dépendant de IBE ou RESET.
Remarque également que la non rep. grand est toujours
trouvé à droite l'ini avec IBE et l'up avec IBE.

On peut également employer les instructions ISET et AGEI avec des variables nommées non allouées à un emplacement de fichier : à cette fin, il est impératif dans ce cas de remplir la variable de réception $\langle X \rangle$ contenant déjà une chaîne de caractères nulle de la longueur voulue.

Example:

```

10 A# = "TELEPHONE"
20 B# = "JEAN- MARIE"
30 C# = "RENE"
40 LIBET A# = C#
50 RIET B# = C#
60 PRINT A# ; " "
70 PRINT B# ; " "
80 END

```

RUN
 RENE.
 RENE.
 etc.

3.5.4 MAKE PEDUNGS

① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯ ⑰ ⑱ ⑲ ⑳ ㉑ ㉒ ㉓ ㉔ ㉕ ㉖ ㉗ ㉘ ㉙ ㉚ ㉛ ㉜ ㉝ ㉞ ㉟ ㊱ ㊲ ㊳ ㊴ ㊵ ㊶ ㊷ ㊸ ㊹ ㊺ ㊻ ㊼ ㊽ ㊾ ㊿

Les valeurs numériques nous font constater que les
variations ne peuvent pas être prises telles quelles.

[illegible]

Supposons que nous ayons à nouveau la valeur de F_i dans un fichier à accès direct. Nous pourrions résoudre le problème

```
10 OPEN 'TEST' AS #1 EN * 17
20 FI = #1:11111111
30 F10 = 11001111
40 FIELD #1, 17 AG N#
50 LSET N# = F10
60 PUT #1,1
70 CLOSE
80 END
```

Le procédé que nous venons d'employer est universel : n'importe quel son intervenant dans la formation d'un mot, nous pouvons le remplacer, il faut, en effet, 17 caractères différents pour écrire un mot en français. On peut donc représenter le son d'un caractère par un caractère, par exemple, on peut représenter le son "a" par le caractère "a", le son "b" par le caractère "b", etc.

Or nous avons guère valeur en double jennine, est
nécessaire en RMH ne nuit ainsi à l'écologie pour les miniers
n'importe où, et de ce fait pour les miniers en ligne, le
pourrait de la l'œuvre en valeur de FI nous en même
forme? C'est la naissance d'un genre de nouvelle qui réalise
la transition.

```
10 OPEN "TEST" AS #1 | EN UN
20 PI = AIN(1)100
30 FICID #1, @ NO NN
40 LOST NO = FICID(PI)
50 PUT #1,1
60 CLOSE
70 END
```

Ordon à cette fonction, nous avons égaré l'information de la ligne 30 du premier programme mais nous avons réussi à trouver dans la fin du dictionnaire un valeur numérique de 17 nouvelles typographiques en seulement 8 octets. Il devient dès lors possible de planer dans un environnement de 384 octets, 32 octets supplémentaires doublement.

Pour rassembler, voici le format pour lesquels les différents
volontaires municipaux ont travaillé.

U.S. Department of Energy

7	6	5	4	3	2	1	0	10	13	12	11	10	9	8
2	2	2	2	2	2	2	2	6	3	2	2	2	2	2

Low content

Zona ortul

La solution se fait en binaire sur 15 bits. La solution est

L'enregistrement sera l'adressé.

Ne vous attendez pas à la disquette au tourné au premier PUT que vous effectuerez. En effet, le tampon mémoire du fichier est d'abord plein dans le tampon SECTEUR avant d'être écrit sur la disquette.

Exemple :

```
5 MAXFILES=1
10 OPEN 'EXEMPLE.TST' AS TI LEN=29
15 FIELD TI, 2 AS LNM, 27 AS NLS
20 FIELD #1, 15 AS Z14, 2 AS Z27, 1 AS Z37, 8 AS Z77
30 INPUT 'NOM' : N1
35 IF NT = 'TTTT' GO TO 130
40 INPUT 'AGE' : A1
50 INPUT 'SALAIRE' : S1
100 INPUT 'No TEL' : T1
70 GET Z17 = N1
80 GET Z27 = M1 : A1
90 GET Z37 = M1 : S1
100 GET Z77 = N1 : D1
110 I = 1 : PUT TI, I
120 GO TO 30
130 GET LRT = N1 : I1
140 PUT TI, I
150 END
```

Ce programme vous permet de créer et de remplir un petit fichier contenant une chaîne enregistrée de nos jours, son âge, son salaire et son numéro de téléphone. Lorsque vous aurez arrêté l'enregistrement du fichier, il y aura 5 enregistrements (TTTTT) le fichier sera alors fermé et on revient à l'indicateur du BASIC. L'avantage de ce petit exemple est qu'il montre la manipulation de tous les types de variables (chaîne, entier, temps et double précision). Remarque : tout ce qui est premier enregistrement est réservé pour indiquer combien d'enregistrements ont été écrits dans le fichier à partir du point numéro 2. En effet, au moment de l'arrêt de l'enregistrement, la variable I indiquant le numéro du dernier enregistrement, est écrite dans le premier record dans la variable IRT dans le fichier.

3.3.6 GET

GET [#<numéro fichier>] I, <X>

L'instruction GET lit l'enregistrement numéro <X> du fichier ouvert sous le nom de fichier I dans le tampon mémoire réservé à ce fichier. De ce fait, toutes les variables définies par l'instruction FIELD sont remplies avec les données provenant de l'enregistrement.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui indique sur quel fichier porte l'instruction. C'est l'instruction OPEN qui associe un numéro au nom du fichier. Il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers

ouvert par MAXFILES avec de la façon un maximum de 6.

<X> est une constante, une variable ou l'expression numérique indiquant quel enregistrement du fichier doit être lu. <X> ne peut être inférieur à 1 ni supérieur à 1,297,967,295. Ce nombre est, bien entendu, théorique car il dépend en fait du nombre d'enregistrement qui ont été écrits dans le fichier par les instructions PUT.

Lorsque <X> est nul, la lecture se fera sur l'enregistrement qui est le dernier enregistrement enregistré par GET ou PUT pour ce fichier, ou alors sur le premier enregistrement si le GET ou PUT n'a encore été utilisé.

Exemple : Récupérer le fichier après dans le fichier PUT.

```
10 MAXFILES = 1
20 OPEN 'EXEMPLE.TST' AS TI LEN = 29
30 FIELD TI, 2 AS LNM, 27 AS NLS
40 FIELD #1, 15 AS Z14, 2 AS Z27, 1 AS Z37, 8 AS Z77
50 GET I1, 1 : I1 = CVI(I1)
60 FOR I=2 TO I1
70 GET #1
80 PRINT 'NOM' : Z17
90 PRINT 'AGE' : CVI(Z27)
100 PRINT 'SALAIRE' : CVI(Z37)
110 PRINT 'No TEL' : CVI(Z77)
120 NEXT I
130 CLOSE
140 END
```

Les lignes 30 à 110 sont le premier record pour obtenir dans la variable I1 le nombre d'enregistrements de ce fichier. Les lignes 40 à 120 traitent une boucle qui lit et affiche les records 2 à I1. Remarque : que la DCL de la ligne 70 n'a donc pas besoin de <numéro d'enregistrement> car il lit les records en séquences.

On peut aussi obtenir le contenu du tampon mémoire par les instructions INPUT et LINE INPUT, mais dans ce cas, le contenu du tampon mémoire doit contenir les marques de séparation reconnues par l'instruction INPUT.

Exemple :

```
10 OPEN 'TEEL.TST' FOR OUTPUT AS TI
20 PRINT #1, 'BONJOUR,'
30 AS '1' : P1=123456 : C1=12345678901237
40 PRINT #1, A1, B1, C1
50 CLOSE
60 OPEN 'TEEL.TST' AS I
70 GET #1
80 INPUT #1, A1, B1, C1
90 PRINT A1, B1, C1
100 CLOSE
110 END
```

3.3.7 Conversion d'une chaîne en valeurs numériques

CVI (X\$): Convertit en Entier (I=Integer)
CVS (X\$): Convertit en Simple précision (S=Simple)
CVD (X\$): Convertit en double précision (D=Double)

Les fonctions CVI CVS CVD convertissent une variable chaîne extraite du langage d'un fichier à accès direct en une valeur numérique entière (CVI), simple précision (CVS) ou double précision (CVD).

la chaîne (X\$) doit avoir une longueur de 7 octets pour CVI, 4 octets pour CVS et 8 octets pour CVD. Plus généralement, on peut dire que ces fonctions convertissent des données numériques du format sous lequel elles résident en mémoire ou dans le langage d'un fichier à accès direct en valeurs numériques.

Exemple:

```
10 A$ = MKI(1024)
20 B$ = MKS(123456)
30 C$ = MKD(12345678901234)
40 PRINT CVI(A$)
50 PRINT CVS(B$)
60 PRINT CVD(C$)
70 END
```

L'exemple ci-dessus montre bien que les fonctions CVI CVS CVD réalisent la fonction inverse de celle de MKI MKS MKD (voir le chapitre 3.5.4).

Exemple:

```
10 MAXFILES = 1
20 OPEN "EXEMPLE.BI" AS #1 LEN = 29
30 FIELD #1, 15 AS W1$, 2 AS W2$, 4 AS W3$, 8 AS W4$
40 GET #1, 1
50 PRINT "NOM: " W1$
60 PRINT "AGE: " CVI(W2$)
70 PRINT "SALAIRE: " CVS(W3$)
80 PRINT "No TEL: " CVD(W4$)
90 CLOSE : END
```

3.5.8 Fermeture d'un fichier à accès direct

CLOSE {#(numéro fichier) | ...}

L'instruction CLOSE conclut les opérations d'entrée/sortie sur un fichier.

(numéro fichier) est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier; il faut donc reproduire (et le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction s'applique. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

Une instruction CLOSE peut fermer plusieurs fichiers à la fois. Indiquez simplement les numéros de fichier désirés, séparés par une virgule. Si CLOSE ne contient aucun numéro de fichier, elle fermera tous les fichiers ouverts.

L'association entre un nom de fichier et son numéro se termine dès la fin du CLOSE. Le fichier peut dès lors être ré-ouvert sous le même numéro ou sous un autre numéro. De même, le numéro de fichier pourra dorénavant servir pour ouvrir n'importe quel fichier.

L'instruction CLOSE est vitale si vous avez procédé à des écritures d'enregistrement car le tampon mémoire final ne sera écrit sur la disquette qu'à ce moment. En plus, le FAT et l'autre du Directory correspondront à ce fichier seront écrites sur la disquette pour refléter un éventuel agrandissement de votre fichier et indiquer la date et l'heure de cette modification.

Les instructions END, CLEAR et NEW provoquent également la fermeture de tous les fichiers ouverts.


```

190 PRINT "Retour pour continuer" : GOTO pour aller"
200 GO = INKEY : IF GO="" GOTO 200
210 IF GO=<CHR(113) GOTO 80
220 IF GO=<CHR(127) GOTO 200
230 CLOSE
240 CLS
250 END
260 IF ERR=70 GOTO 280
270 ON ERROR GOTO 0
280 PRINT "PAS DE DISQUE DANS LE LECTEUR"
290 INPUT "Quelques quand problème corrigé" : GO
300 CLS : RESUME

```

Si vous avez exécuté et essayé ce programme, vous trouverez certainement qu'il est beaucoup plus rapide que son équivalent en fichier séquentiel surtout s'il comporte beaucoup d'enregistrements, mais vous devez noter qu'il est limité à la recherche de l'adresse de quelqu'un en entrant son nom puis qu'il donnera son numéro d'enregistrement qu'il est difficile de saisir pour chaque nom...

C'est ici qu'intervient ce que l'on appelle l'accès aléatoire. Si vous voulez vraiment utiliser le programme précédent, vous devez également avoir de vous constituer une liste aléatoire de noms d'enregistrements pour que les recherches dans le fichier soient plus rapides. Vous pouvez rapidement l'adresser de M. Dupont, vous constituer d'abord votre liste à "Détail" et utiliser le numéro d'enregistrement écrit et regardé pour demander à l'ordinateur le reste des informations.

Mais pourquoi ne pas faire exécuter le programme par l'ordinateur? Deux possibilités s'offrent à nous.

On peut créer un tableau résident en associant avec chaque nom le numéro d'enregistrement ou l'adresse des données résidentes. Ce tableau devrait être constitué au démarrage du programme ou à l'usage des enregistrements pour en extraire uniquement le nom et le numéro dans le tableau.

On pourrait également créer un deuxième fichier contenant le nom de la personne et le numéro de l'enregistrement du fichier principal où réside le reste des informations. Ce fichier étant très court, il ne gênerait pas trop la recherche de l'information.

Le premier système n'est valable que si la possibilité existe de placer dans les noms et adresses conjointement avec le programme maître les données du second système, un peu plus long il est vrai, convient tout de même très bien. Nous nous concentrerons sur le deuxième système dans le but d'exercer devant vous notre apprentissage du Disk Basic.

La méthode la plus rationnelle consiste à écrire un petit programme séparé qui va créer ou modifier le fichier Index. Ainsi, chaque fois que le fichier principal aura subi des modifications, il faudra de lancer ce petit programme pour réactualiser le fichier Index.

```

10 CLEAR 500 : MAXFILES = 2
20 OPEN "ADRESSE.RND" AS #1 LEN = 91
30 OPEN "ADRESSE.LNK" AS #2 LEN = 18
40 FIELD #1, 2 AS LNR, 89 AS NLR
50 FIELD #1, 14 AS I1, 75 AS NLR
60 FIELD #2, 14 AS V1
70 GET #1,1
80 LR = CVT(LNR)
90 FOR I=1 TO LR
100 GET #1,1
110 I1 = V1 : I2 = I1
120 PUT #2,1
130 NEXT
140 CLOSE
150 END

```

En fait, ce programme ne fait que recopier dans le fichier ADRESSE.LNK les données au début de chaque enregistrement de fichier ADRESSE.RND sous la forme d'un enregistrement.

Voilà maintenant la seconde partie du programme de consultation sur cette liste-ci va nous permettre de chercher l'adresse d'une personne par son nom et de le voir son numéro.

```

10 CLEAR 500 : MAXFILES = 2
20 ON ERROR GOTO 290
30 OPEN "ADRESSE.RND" AS #1 LEN = 91
40 OPEN "ADRESSE.LNK" AS #2 LEN = 18
50 FIELD #1, 2 AS LNR, 89 AS NLR
60 FIELD #1, 14 AS I1, 75 AS NLR, 25 AS I2, 6 AS I3
70 FIELD #2, 14 AS V1
80 GET #1,1
90 LR = CVT(LNR)
100 CLS
110 PRINT "QUEL NOM CHERCHER-VOUS?" : NO
120 FOR I=1 TO LR
130 GET #1,1
140 IF V1 = NO GOTO 180
150 NEXT
160 PRINT "Désolé, ce nom n'existe pas dans la liste"
170 GOTO 130
180 GET #1,1
190 PRINT : PRINT I1 : PRINT I2
200 PRINT I3 : PRINT I4
210 PRINT CVT(I25) : PRINT I26
220 PRINT CVT(I27)
230 PRINT
240 PRINT "Retour pour continuer" : GOTO pour aller"
250 GO = INKEY : IF GO="" GOTO 250
260 IF GO=<CHR(113) GOTO 100
270 IF GO=<CHR(127) GOTO 230
280 CLOSE : END
290 IF ERR=70 GOTO 310
300 ON ERROR GOTO 0
310 PRINT "Pas de disque dans le lecteur"
320 INPUT "Retour quand problème corrigé" : GO
330 CLS : RESUME

```

En pratique, la technique du fichier à accès direct Index

est un peu plus sophistiqué qu'un dictionnaire. On va, en effet, classer les noms du fichier INDEX par ordre alphabétique et la recherche ne se fera pas séquentiellement comme c'est le cas ici, mais par dichotomie pour réduire le nombre d'accès au minimum.

3.6.3 Mise à jour d'un fichier

La tenue d'un carnet d'adresses ne se limite pas à enregistrer les données dans le fichier et ensuite à consulter ce fichier. Il arrive fréquemment que des ajouts ou des modifications aux données déjà enregistrées soient nécessaires. Le programme suivant vous permet de contourner ce problème.

```

10 CLEAR 500 : MAXFILES = 2
20 ON ERROR GOTO 290
30 OPEN "ADRESSE.RND" AS #1 LEN = 71
40 OPEN "ADRESSE.INX" AS #2 LEN = 16
50 FIELD #1, 2 AS LR$, 69 AS NUL$
60 FIELD #1, 16 AS Z1$, 16 AS Z2$, 25 AS Z3$, 6 AS Z4$
  4 AS Z5$, 16 AS Z6$, 8 AS Z7$
70 FIELD #2, 16 AS Y1$
80 GET #1,1
90 LR = CVI(LR$)
100 CLS
110 PRINT "QUEL NOM CHERCHER-VOUS?"; N$
120 FOR I=2 TO LR
130 GET #2,1
140 IF Y1$ = N$ GOTO 180
150 NEXT
160 PRINT "Ce nom n'existe pas. Voulez-vous l'insérer?"
161 INPUT A$: IF LEFT$(A$,1) = "N" GOTO 230
162 IF LEFT$(A$,1) <> "O" GOTO 160
163 LR = LR+1
164 INPUT "PRENOM"      "JP$
165 INPUT "RUE"         "JR$
166 INPUT "NUMERO"      "JN$
167 INPUT "CODE POSTAL" "JCP$
168 INPUT "VILLE"      "JV$
169 INPUT "TELEPHONE"   "JT$
170 LSET Z1$ = N$
171 LSET Z2$ = P$
172 LSET Z3$ = R$
173 LSET Z4$ = NO$
174 LSET Z5$ = MK$(CP)
175 LSET Z6$ = V$
176 LSET Z7$ = MK$(T$)
177 PUT #1,1
178 LSET Y1$ = N$
179 PUT #2,1 : GOTO 230
180 GET #1,1
190 PRINT : PRINT Z1$ : PRINT Z2$
200 PRINT Z3$ : PRINT Z4$
210 PRINT CV$(Z5$) SPC(5) Z6$
220 PRINT CV$(Z7$)
221 PRINT : PRINT
222 INPUT "NOM"         "JN$
223 GOTO 164
230 PRINT

```

```

240 PRINT "Return pour continuer - ESC pour arrêter"
250 AS=INKEY$: IF AS="" GOTO 250
260 IF AS=CHR$(13) GOTO 100
270 IF AS<>CHR$(27) GOTO 250
280 CLOSE : END
290 IF ERR=70 GOTO 310
300 ON ERROR GOTO 0
310 PRINT "Pas de disque dans le lecteur"
320 INPUT "Return quand problème corrigé"; A$
330 CLS : RESUME

```

3.7.1 BASIC Disk Free Space

DISK<Numéro lecteur>

La fonction BASIC permet d'obtenir le nombre de secteurs libres sur la disquette insérée dans le lecteur numéro X. Pour rappel, un secteur est l'unité logique d'allocation d'espace sur disquette. Tous les types de disquettes de l'annexe A ont une taille de secteur de 512 octets (512 bytes) et les types PC et FD ont la taille de secteur de 512 octets.

Numéro lecteur > est un nombre de 0 à 9 indiquant à quel lecteur vous souhaitez obtenir l'espace libre. 0 veut pour le lecteur courant, 1 pour le lecteur A1, 2 pour le lecteur A2 et ainsi de suite jusqu'à 9 pour le lecteur I9.

Si vous désirez obtenir l'espace libre du lecteur en nombre d'octets, multipliez simplement le nombre obtenu par 1024 ou 512 suivant le type de disquette insérée. Rappelez-vous également que le plus petit programme ou fichier occupe toujours au moins 1 cluster sur la disquette.

Exemple:

```
PRINT DISK101 : Donne le nombre de secteurs libres du
```

```
lecteur courant
```

```
PRINT DISK11:1024 : Donne le nombre d'octets libres du
```

```
lecteur A1
```

```
360 IF NOT (0)10 THEN PRINT "PAS D'ESPACE SUR LA DISQUE"
```

3.7.2 BASIC Disk Input

DISK<Numéro lecteur>,<Numéro secteur>

La fonction BASIC permet de lire le secteur partiel le (numéro secteur) de la disquette insérée dans le lecteur pointé le (numéro lecteur) vers le lecteur désigné du Directory dont l'adresse est donnée par le pointeur F3511 sur le bit de lecture.

Numéro de lecteur > est un chiffre de 0 à 9 où 0 veut pour le lecteur courant, 1 pour le lecteur A1, 2 pour le lecteur A2 et ainsi de suite jusqu'à 9 pour le lecteur I9.

Numéro secteur > est un nombre indiquant quel secteur doit être lu. Le premier secteur de la disquette porte toujours le numéro 0 et le secteur dépend du type de disquette. Pour l'annexe A pour connaître pour chaque type de disquette son nombre de secteurs. Ainsi, le type de disquette FS 13"4 - Simple Face - 80 pistes - 9 secteurs/piste contient 720 secteurs numérotés de 0 à 719. Les 9 premiers secteurs 10 à 81 de cette disquette se trouvent sur le piste 0, les 9 suivants 19 à 171 sur le piste 1 et ainsi de suite. Pour les disques double face, les

9 premiers secteurs se trouvent sur le piste 0 de la face 1, les 9 suivants sur la piste 0 de la face deux, les 9 suivants sur la piste 1 de la face 1 et ainsi de suite.

Pour connaître l'adresse mémoire du langage Directory où sera effectué le secteur pointé par la formule suivante:

ADRESSE = PEEK(16F3511 + 256 * PEEK(16F3521)

ATTENTION! Il faut exploiter le contenu du langage Directory avant qu'un programme comme LOAD, RUN, MERGE, KILL, SAVE ou OPEN ne le modifie.

Exemple: le secteur 0 se trouve sur la piste 0 et 10 veut pour le secteur 10. L'appelle l'adresse de la disquette de la disquette de la version de ce ROM. Ce code programme se veut permettre de connaître quel secteur est inséré le disquette présente dans le lecteur 0:

```
10 CLS
20 NI = PEEK(16F3511) + 256 * PEEK(16F3521)
30 @=PEEK(NI),0
40 @DR I=NI*3 TO NI+10
50 @PRINT CHR$(PEEK(NI))
60 NEXT I
70 END
```

Le variable @ reçoit une chaîne de caractères. Elle est placée le quelconque parce que BASIC est une fonction et son une chaîne de caractères.

3.7.3 BASIC Disk Output

DISK<Numéro lecteur>,<Numéro secteur>

L'écriture BASIC permet d'écrire le langage mémoire réservé au Directory et indique par le pointeur F3511 sur le secteur (numéro secteur) de la disquette insérée dans le lecteur (numéro lecteur).

Numéro de lecteur > est un chiffre de 0 à 9 où 0 veut pour le lecteur courant, 1 pour le lecteur A1, 2 pour le lecteur A2 et ainsi de suite jusqu'à 9 pour le lecteur I9.

Numéro secteur > est un nombre indiquant quel secteur doit être écrit. Le premier secteur de la disquette porte toujours le numéro 0 et le secteur dépend du type de disquette. Pour l'annexe A pour connaître pour chaque type de disquette son nombre de secteurs. Ainsi, le type de disquette FS 13"4 - Simple Face - 80 pistes - 9 secteurs/piste contient 720 secteurs numérotés de 0 à 719. Les 9 premiers secteurs 10 à 81 de cette disquette se trouvent sur la piste 0, les 9 suivants 19 à 171 sur le piste 1 et ainsi de suite. Pour les disques double face, les 9 premiers secteurs se trouvent sur la piste 0 de la face 1, les 9 suivants sur la piste 0 de la face deux, les 9 suivants sur la piste 1 de la face 1 et ainsi de suite.

Pour connaître l'adresse mémoire du tampon Directory où sera transférée le secteur, appliquez la formule suivante:

ADRESSE = PEEK(4HF351) + 256 * PEEK(4HF352)

Attention que cette instruction peut être une vraie diquette puisque elle y écrit directement sans aucun contrôle. Choisissez donc une zone de 65 diquettes où vous êtes sûr qu'il n'y a pas de fichier. Pour vos essais, il est d'ailleurs préférable d'employer une diquette nouvellement formatée et qui ne contient aucun fichier. Attention aussi à la zone réservée (Boot secteur - FAT - Directory - voyez le chapitre 4).

Si le secteur 1 indiquait également le type de diquette est détruite, cette instruction ne peut pas fonctionner.

Exemple: Programme de copie physique d'une diquette entière, bien que ce programme puisse tourner sur les configurations à 1 ou 2 disquettes, il est recommandé de l'utiliser sur des MSX à deux unités, sinon vous devrez interchanger les diquettes extérieurement de façon qu'il y a de lecture à copier. Le nombre de secteurs à copier est obtenu par la lecture des octets 19 et 20 du secteur 0 du de la diquette à copier (ligne 50 = 701).

```
10 CLS
20 PRINT "COPIE DU LECTEUR A1 SUR LE LECTEUR B1"
30 INPUT "PRET D/N?":N%
40 IF LEFT(N%,1)<>"0" GOTO 30
50 A% = DISK(1,1)
60 B% = PEEK(4F351) + 256 * PEEK(4F352)
70 MS = PEEK(4F19) + 256 * PEEK(4F20)
80 FOR I=0 TO MS
90 A% = DISK(1,I)
100 DISK(2,I)
110 PRINT I
120 NEXT I
130 END
```

3.7.4 VARPTR VARIABLE Pointer

VARPTR(Numéro fichier)

La fonction VARPTR(%) recherche l'adresse mémoire où est implémenté le File Control Block (FCB) du fichier (numéro fichier).

Pour une explication détaillée du FCB, voyez le chapitre 5 mais, en résumé, on y trouve le nom du fichier; le date et l'heure des dernières modifications; le dimension du fichier, le premier cluster du fichier; le dernier cluster affecté; la longueur du record, etc...

Ainsi, par exemple, la position FCB:25 indique en quelle position le fichier se trouve dans le Directory. Le petit programme suivant va vous indiquer, pour un fichier au choix, la position qu'il occupe dans le Directory.

```
10 CLS
20 INPUT "NOM DU FICHIER":F$
30 OPEN F$ AS #1
40 FCB = VARPTR(1)
50 P = PEEK(FCB+25)
60 PRINT "POSITION":P
70 CLOSE
80 END
```

3.7.5 Messages d'erreur du Disk-Basic

Lorsqu'une erreur intervient dans le déroulement des instructions d'un programme, celui-ci s'interrompt et un message approprié est affiché.

Le Disk Basic a ajouté une série de nouveaux messages d'erreur dont le liste suit. Grâce à l'instruction ON ERROR GOTO, et à l'instruction IF ERR=XX THEN, on peut éviter l'arrêt du programme et traiter soi-même l'erreur.

- 50 FIELD OVERFLOW
Une instruction FIELD tente d'alloquer plus d'espace à une variable qu'il n'en a été prévu dans l'OPEN du fichier à accès direct concerné.
- 51 INTERNAL ERROR
Erreur interne au Disk Basic qui est tellement rare que MICROSOFT demande qu'on lui rapporte les circonstances dans lesquelles elle se produit.
- 52 BAD FILE NUMBER
Une instruction ou une fonction fait référence à un fichier qui n'est pas ouvert, ou encore le numéro de fichier est hors tolérance plus grand que le ou plus grand que le maximum prévu par l'instruction MAXFILES.
- 53 FILE NOT FOUND
Un LOAD, SLOAD, RUN, MERGE, NAME ou KILL fait référence à un fichier qui n'existe pas ou le fichier demandé.
- 54 FILE ALREADY OPEN
Une instruction OPEN est faite pour un fichier qui est déjà ouvert, ou une instruction KILL essaye de détruire un fichier qui n'est pas encore fermé.

L'organisation de la disquette

4.1 Le découpage de la disquette

Une disquette MSX est découpée en 5 zones:

- 1) Le boot sector
- 2) La FAT (File Allocation Table)
- 3) Le copie de la FAT
- 4) Le directory
- 5) La zone des fichiers

Suivant le type de disquette, il y a une ou deux faces, 40 ou 80 pistes, 8 ou 9 secteurs par piste. La face 1 se trouve du côté du moyeu d'entraînement de la disquette, la face 2 du côté de l'étiquette. La piste 0 est située près du bord de la disquette, tandis que la dernière piste est située près du centre.

Le secteur 0 d'une piste est situé le premier après l'index qui est un repère physique indiquant l'endroit du début de la piste. Les autres secteurs sont placés séquentiellement par ordre numérique.

La numérotation des secteurs se fait en donnant le numéro 0 au premier secteur de la première piste de la première face. On attribue ensuite les numéros suivants aux autres secteurs de la piste. Lorsque tous les secteurs d'une piste ont reçu leur numéro, la numérotation continue sur la face 2 pour les disques double face. Pour les disquettes simple face, ou lorsque tous les secteurs de la face 2 piste 0 ont reçu leur numéro pour les disquettes double face, on passe à la piste suivante et ainsi de suite.

L'emplacement de chacune des 5 zones dépend du type de disquette (voir l'annexe A). Cependant, la plupart des utilisateurs emploient des disquettes de 3 1/2 soit simple face (360K) ou double face (720K).

Voici un tableau indiquant les secteurs réservés à ces 5 zones.

SECTEURS	DESCRIPTION
0	0! Boot sector MSXDOS. Permet l'installation du MSXDOS!
1	1! MSXDOS.SYS est présent sur la disquette.
2	2! FILE ALLOCATION TABLE (FAT) : Ces secteurs contiennent une table d'allocation des secteurs pour les différents fichiers de cette disquette.
3	3! 4! Copie descendante de la F.A.T.
4	4! 5! C'est une copie des secteurs précédents
5	5! 7! Ces secteurs forment la répertoire (Directory).
6	6! 8! C'est la liste des noms de fichiers contenus
7	7! 9! dans cette disquette avec, pour chacun d'eux,
8	8! 10! une série de paramètres
9	9! 11!
10	10! 12!
11	11! 13!
12	12! 14! Secteur à partir duquel vos fichiers s'installent
1719/1439	1719/1439! C'est le dernier secteur de votre disque.

4.2 Le cluster

Littéralement, cluster signifie collection, agglomération. Dans notre cas, il représente un ensemble de secteurs sur disquette.

Le norme MSX a choisi une taille de cluster dépendant de la capacité de la disquette (voir annexe A1. Comme la plupart des disquettes MSX ont deux secteurs par cluster et que la taille habituelle d'un secteur est de 512 octets, on dit qu'un cluster vaut 1K. Mais rappelez-vous que l'on pourrait, théoriquement, voir venir sur le marché d'autres types de lecteurs de disquettes avec une taille-secteur différente et un nombre de secteurs par cluster différent lui aussi. En conséquence, les concepteurs du DOS (Disk Operating System) ont cherché à s'écarter de la notion de secteur en créant le cluster. Cela leur permet d'écrire leur DOS sans aucune contrainte de matériel.

Rappelons ici que le secteur est la plus petite information qui puisse être lue ou écrite sur disque. Imposable, par exemple, de lire ou d'écrire 2 caractères de sur le disque.

Lorsque le Disk-Basic exécute un programme sur disquette, il va allouer à ce programme non pas le nombre de secteurs strictement nécessaires mais plutôt autant de clusters que le résultat de la division de la longueur du programme par la longueur du cluster, plus un cluster.

Exemple Taille K Nombre de Clusters

Programme1	217	0 1
Programme2	649	0 1
Programme3	1024	1 2
Programme4	1739	1 2
Programme5	5120	5 6

Avantage du système:

Il est réservé plus d'espace que nécessaire sur le disque. Dès lors, si vous modifiez ce programme de telle sorte qu'il s'agrandisse, il y a déjà une place réservée sur le disque pour cela et il n'est pas nécessaire de procéder à des extensions de cluster sur la disquette. Donc tout le programme se trouve concentré à la même place sur la disquette; d'où un accès plus rapide aux informations.

Inconvénient du système:

Si vous sauvez beaucoup de petits programmes sur votre disquette, à chaque fois il sera réservé un espace plus grand et dès lors les 360K de votre disque ne pourront pleinement être utilisés.

4.3 Le record

Le record (enregistrement) est l'unité logique d'information à transférer du ou vers le fichier, aux yeux du programmeur.

En base, cette notion ne s'applique qu'aux fichiers à accès direct et se taille est programmable de 1 à 256 octets. La limitation de la taille à 256 octets est due au fait qu'il faut réserver des espaces en mémoire pour chaque fichier ouvert, sans trop étendre pour autant cette mémoire réservée.

En MSX-DOS, le record a une largeur de 128 octets pour toutes les fonctions compatibles CP/M (fonctions 0F à 24 - voir chapitre 7). Pour les fonctions typiquement MSX-DOS (26H et 27H), la longueur du record est programmable par l'utilisateur de 1 à 65536 octets (la dernière valeur est théorique) tout dépend de la place mémoire réservée au temps du record.

4.4 L'extent

Le programmeur Base peut ignorer ce terme car il n'a de signification que pour certaines fonctions MSX-DOS compatibles CP/M. En CP/M, le terme extent n'existe pas. L'unité d'allocation sur le disque est l'extent. Il occupe des secteurs consécutifs sur le disque et sa taille est un des paramètres du CP/M. Supposons qu'il ait une taille de 16K, il pourrait alors stocker 128 records de 128 octets. $128 \times 128 = 16384 = 16K$.

Lorsque le programmeur CP/M veut lire ou écrire un ou plusieurs records d'un fichier séquentiel, il peut préciser dans une zone mémoire spéciale appelée FCB (File Control Block) à partir de quel record, de quel extent, il veut manipuler son fichier séquentiel. En effet, il est réservé 16 bits dans le FCB pour donner le numéro d'extent et 7 bits pour le numéro de record dans cet extent. Ce terme est défini ici pour assurer la compréhension du paragraphe concernant le FCB (voir le chapitre 5); à part cela, il n'a donc aucun sens en MSX-DOS.

Le terme boot sector vient de l'expression anglaise BOOTSTRAP, que l'on pourrait traduire par passerelle. Tout comme la passerelle d'un bateau nous permet de le charger, le bootstrap sector (ou abrégé boot sector) permet de charger le DOS (Disk Operating System = Système d'exploitation du disque) dans la mémoire de l'ordinateur. Chaque ordinateur dispose d'un petit programme intégré dans son ROM qui va provoquer, à l'allumage de l'ordinateur, la lecture du secteur 0 de la disquette insérée dans le premier lecteur du système.

Le contenu de ce secteur 0 est lui-même un programme qui va s'occuper de charger le fichier du système d'exploitation proprement dit en mémoire.

En plus de ce programme, le boot sector contient une série de paramètres sur les caractéristiques physiques de la disquette et sur l'organisation des données sur celle-ci. MICROSOFT, qui a écrit le MSX-DOS pour MSX et le MS-DOS pour IBM ne s'est pas compliqué la tâche, puisque le boot sector des deux systèmes est totalement identique, à l'exception, bien sûr, du programme de chargement qui est écrit en langage machine Z80 pour le MSX-DOS et en langage machine 8086 pour le MS-DOS.

Les 256 premiers caractères du boot sector sont réservés en mémoire à l'adresse 0000H pour le fichier d'initialisation du système.

En résumé, le boot sector est chargé lorsque vous introduisez la disquette dans votre lecteur immédiatement après le démarrage de l'ordinateur. Les 256 premiers caractères du secteur se chargent à l'adresse mémoire 0000H. Le tableau caractéristique du disque est chargé en mémoire, mais le système MSX ne l'utilise pas du tout (il se base sur le premier caractère de la FAT pour identifier le type de votre disque et en déduire ses caractéristiques lui-même). Si le fichier MSDOS.SYS est présent sur votre disquette, il sera chargé et installé le MSX-DOS en mémoire, chargera COMMAND.COM et lui passera le main. Vous verrez alors apparaître le prompt directionnel du MSX-DOS: A>

Voici le démarrage du secteur 0 :

[illegible]

4.7 La F&T (File Allocation Table)

L'analyse du fonctionnement de la FRI se vous initier e la deluxa employés par le HSI et par l'ISM PCI pour sauver vos liliers sur disquette.

Suffisamment pour trier les choses, que vous senez de forder une équerre. Elle se éorlent dans on tre Jules lisher. Grémis en petit programme Bels et sauven-à pas la commande SAVE "PROGRAM.DAT". Le système commença pas étre dans la prealim élie de la éicécy en éon é programme, se éouquas, le dals et l'heurs tel vous éssé un ééééé.

Nous avons déjà vu dans le paragraphe du linéaire, que les données sont sauvegardées sur disque par une telle échantillonnage. C'est, par exemple 12 bits/échantillon pour le signal des types de échantillon. Dans une échantillon de 256K, l'échantillon est 320 échantillon. Une échantillon de 256K, l'échantillon est 320 échantillon. Les échantillons sont numérotés de 0 à 319 et de 0 à 719 ainsi le type de échantillon.

un menu qui à l'échelle centim, entre vos livres, un bon secteur, etc. Les deux de chacun est et une d'ailleurs de 7 secondes. Ce n'est le nombre de livres libre utilisables pour vos livres et édité 425 livres pour une disquette 500K - 61/2" et 6 716 livres pour une disquette de 800K - 3 1/2". Pour des raisons d'organisation interne de la table, le premier cluster d'adresse des livres est le 002.

Les meilleurs état son toujours lui personnel é toute
opérations étre par la région et étre des an n'importe
dans un temps qui leur est personnel étre. Ce temps
est un temps de la ville étre des de celle étre
il faut étre étre et, et étre étre.

Le 980 est divisé en 360 ou 420 séries numériques de 0 à 359/719, l'unité 0 correspond au milieu 0, l'unité 1 correspond à 1... et l'unité 459 au milieu 359.

Chaque entrée de la table contient 12 bits et permet donc d'encoder un nombre de 0 à 4095. Les hexadécimaux 0000 à 000F, le nombre en 01 indique aux listes 01 à 0F les listes correspondantes du même 01 à 0F de la table. Ainsi donc, si on trouve 0001 il se trouve dans l'entrée 1, cela signifie que le cluster 01 est dans la liste 01.

Revenons maintenant à notre petit problème "PRODIGES". Nous allons nous occuper d'abord de la sous-phrase "PRODIGES". Analyse le bloc "S" en le notant par le MEI.

La situation de départ du candidat libéral élu en 1982, dans la première circonscription de l'académie, est la suivante : 20 000 habitants en position de 27 (le candidat libéral). Le système va donc suivre le début du vote à proportionnelle dans la situation 2 et concurrencer de 1043 délégués républicains sur la structure des libéraux.

Si votre projet dure-
ecl plus long que 1075 octets, il
devra contenir un algorithme
puisque'il n'y a plus place
dans la classe 2. Pour ce faire,
le MSX utilise la table 107

de partir du cluster 2 jusqu'à ce qu'il rencontre une entrée libre. Une entrée est libre lorsqu'elle contient 000.

Puisque chaque disque vient d'être formaté, toutes les entrées de la table 2 sont libres. Le système va donc insérer l'entrée 3 dans la table 2. Le système va maintenant écrire la adresse de l'entrée libre dans l'entrée réservée au cluster 2 et écrire les 1024 octets suivants de votre programme dans la cluster 3. Cette technique va ainsi se répéter tant qu'il subsiste une entrée de votre programme dans la table 2.

Quand la dernière position est écrite, la cluster 3 est insérée dans la table de description de votre programme et l'entrée correspondante de ce dernier cluster est marquée avec 000, ce qui signifie que la cluster 3 est la dernière ajoutée à votre programme.

Donc, le PROGRAM1.BAS a une longueur de 6150 octets par exemple, le FAT se compose de :

E Des Description

```
0 F65 Type de chaque lettre vu plus loin
1 000 Cluster réservé lettre vu plus loin
2 003 Le cluster 7 est suivi du 3 et contient les 1024
    premiers octets du programme
3 006 Le cluster 3 est suivi du 6 et contient les 1024
    octets suivants du programme
4 005 Le cluster 6 est suivi du 5 et contient les 1024
    octets suivants du programme
5 006 Le cluster 5 est suivi du 6 et contient les 1024
    octets suivants du programme
6 007 Le cluster 6 est suivi du 7 et contient les 1024
    octets suivants du programme
7 008 Le cluster 7 est suivi du 8 et contient les 1024
    octets suivants du programme
8 00F C'est le dernier cluster et contient les 7 derniers
    octets du programme
9 000 Ce cluster est libre
.. 000 " " "
.. 000 " " "
359 000 Ce cluster est le dernier de la étiquette 12 et
    est libre.
```

Le programme PROGRAM1.BAS sera donc inséré au 1er disque 2, 3, 4, 5, 6, 7 et 8. Au milieu des nombres de secteur, on a des lettres de 1 à 9 qui indiquent la lettre A-I. En effet, la cluster 3 est insérée au secteur 12 pour les étiquettes de 360K et au secteur 14 pour les étiquettes de 720K. Notre programme est donc inséré sur les disques 12 à 25 pour une étiquette de 360K.

Pour chaque disque, que la cluster 8 commence à 24 et 25 et contient que 7 octets de notre programme. Les 1017 derniers octets de la cluster contiennent des données qui ne seront pas lues en temps réel lors d'un chargement de votre programme. Elles prennent du temps de lecture (Sector Buffer) et nous y retournerons donc les 512 octets de dernière section du cluster (général) édités par les 7 derniers octets de votre programme. Ces données se retourneront

sur le secteur 24 et 25.

On peut donc affirmer que la FAT est une table de chargement des données et que chaque cluster est une entrée de la Directory et les autres par le FAT.

Les données insérées dans la table de chargement sont : pour notre programme PROGRAM1.BAS et édités la table de chargement. L'opération de chargement est la même que celle du programme PROGRAM1.BAS excepté que le cluster de votre programme ne sera plus la cluster 2, mais sera trouvé par la recherche d'un cluster libre dans la FAT.

Si notre nouveau programme a une taille de 2000 octets, il sera donc inséré sur les clusters 6 et 10 et le FAT contiendra les entrées suivantes :

```
0 1 2 3 4 5 6 7 8 9 A B C D E...
440-444 003-004-005-006-007-008-009-00A-00B-00C-00D-00E...
PROGRAM1.BAS
PROGRAM2.BAS
```

Mais que va-t-il se passer si nous cherchons cette première entrée pour le modifier et que le résultat de la recherche soit un allongement de notre programme?

Deux possibilités existent : soit la modification a allongé le programme de moins de 1017 octets et alors il reprendra exactement la même place sur le disque ou elle l'a allongé de plus de 1017 octets et alors le programme sera inséré dans la cluster 10 et la modification sera faite sur la cluster 10 et la cluster 11.

Puisque le programme est déjà sur le disque, il est chargé en mémoire. Cette opération consiste à placer en mémoire le programme et à le charger sur le disque. On peut donc dire que le programme est inséré dans la cluster 10 et la cluster 11 et qu'il est inséré dans la cluster 10 et la cluster 11.

Enfin, on peut donc facilement dire, e et d, qu'une recherche du cluster 10 et la cluster 11 est faite et que le programme est inséré dans la cluster 10 et la cluster 11.

Il faut cependant tenir compte que les clusters 7 à 10 ne sont pas libres car ils ont été alloués au programme PROGRAM1.BAS. Si la cluster 6 est allongé de 2000 octets, par exemple, la FAT contiendra donc :

Les lignes 40 à 80 permettent de trouver l'adresse du FCB, du cluster de départ, l'adresse du DFB utilisé, l'adresse de la FAT et le nombre de secteurs par cluster voir chapitre suivant.

La ligne 90 convertit le numéro de cluster en numéro de secteur suivant la formule vue dans ce chapitre.

La ligne 100 affiche le numéro de secteur et la ligne 110 affiche le numéro du deuxième secteur de cluster si la taille de cluster est 2 secteurs.

La ligne 120 est la première des deux formules évoquées plus haut. Elle donne un déplacement à ajouter à l'adresse de la FAT pour trouver l'entrée correspondante au cluster désiré. La ligne 130 donne l'adresse de cette entrée.

La ligne 150 est la deuxième formule évoquée. Elle convertit le contenu des 12 bits de l'entrée de la FAT en numéro de cluster pour les entrées impaires.

La ligne 160 fait de même pour les entrées paires.

La ligne 170 teste si le cluster trouvé n'est pas le dernier de ce fichier.

```
10 CLS:MAXITER=100
20 INPUT "DONNEZ LE NOM DU FICHIER":FS
30 OPEN FS:FOR INPUT AS I
40 FCB=PEEK(FCB+256)+256*PEEK(FCB+384)+128
50 CLU=PEEK(FCB+24)+256*PEEK(FCB+27)
60 DFB=PEEK(DFB+243)+256*PEEK(DFB+241)
70 FAT=PEEK(FAT+119)+256*PEEK(FAT+201)
80 NSC=PEEK(DFB+7)
90 SECT=PEEK(DFB+121)+256*PEEK(DFB+131)+1024*NSC
100 PRINT USING "Secteur ####";SECT
110 IF NSC=2 THEN PRINT USING "Secteur ####";SECT+1
120 DEPLAC=1+(1+(CLU-1)*11)*512
130 INTFAT=FAT+DEPLAC
140 IF CLU=0 GOTO 160
150 CLU=PEEK(INTFAT+DEPLAC)+256*PEEK(INTFAT+DEPLAC+1)
155 GOTO 170
160 CLU=PEEK(INTFAT+DEPLAC)+256*PEEK(INTFAT+DEPLAC+1024)
170 IF CLU=0 GOTO 190
180 END
```

4.8 Structure des fichiers

4.8.1 Le fichier-programme Basic sauve en binaire compressé

Ce fichier a été créé par l'instruction Basic "SAVE ".....". Il contient un indicateur de programme Basic en première position du fichier et est constitué du texte de votre programme Basic tel qu'il se trouve en mémoire.

Vous savez que le texte de vos lignes d'instructions Basic est encodé et que ce que vous voyez à l'écran après un LIST n'est que la traduction de ce qui se trouve réellement en mémoire. Imaginez que vous ayez saisi le petit programme Basic suivant sous le nom de PROGRAM.BAS :

```
10 CLS
20 PRINT"MXS"
30 END
```

Voici ci-dessous le contenu du fichier "PROGRAM.BAS". Le contenu de la première colonne est en hexadécimal.

```
FF Indicateur de programme Basic,
07 Pointeur vers la ligne suivante
00 |
0A | Numéro de la ligne 100
00 | " " "
9F Code de l'instruction CLS
00 Code de fin de ligne
12 | Pointeur vers la ligne suivante
00 | " " " "
14 | Numéro de la ligne 120
00 | " " "
91 Code de l'instruction PRINT
22 |
40 N
33 S
59 X
22 |
00 Code de fin de ligne
1B | Pointeur vers la ligne suivante
00 | " " " "
1E | Numéro de la ligne 130
00 | " " "
81 Code de l'instruction END
00 Code fin de ligne
00 Code fin de programme
00 | " " " "
```

En résumé, le fichier de programme Basic contient la valeur FF comme indicateur de type de fichier suivi du contenu de la mémoire à partir de l'adresse 800h et jusqu'à, et y inclus, l'adresse de l'indicateur de fin de programme (00-00). En réalité, l'adresse de départ dépend du pointeur FCB.

4.8.2 Le fichier de programme Basic sauve en ASCII

Le fichier est créé par l'instruction Basic "SAVE", "A" il contient en ASCII le texte de votre programme Basic tel qu'il apparaît à l'écran après un LIST. Chaque ligne est terminée par le paire de caractères CR-LF (Carriage Return et Line Feed) et est en Hexa. Il n'y a donc aucun indrétil de type de fichier ASCII par contre, le fichier se termine par une marque de fin de fichier que le code hexadecimal 1A. Si vous sauve le programme de l'exemple précédant en ASCII, le fichier contiendra :

```
31-30-20-43-10-53-00-0A
3 0 C L B
32-30-20-50-52-49-4E-54-22-40-53-58-22-0D-0A-
2 0 F R I N T " M B X "
```

```
33-30-20-15-1E-4E-0D-0A-1A --> Marque de fin de fichier
3 0 E M D
```

4.8.3 Le fichier BINAIRE ou langage machine

Il est créé par l'une des deux instructions Basic suivantes :

```
SAVE ".....",Mxxxx,Myyyy,Bxxxx
SAVE ".....",Mxxxx,Myyyy,B
```

Dans le premier cas, il s'agit de sauve une portion de mémoire comprise entre les adresses Mxxxx et Myyyy qui contiennent généralement un programme en langage machine. L'adresse Mxxxx permet de définir à quel endroit l'exécution doit commencer.

Dans le deuxième cas, il s'agit de sauve une portion de la mémoire réservée au VDP (Video Processor) qui contiennent généralement une image.

Dans les deux cas, le contenu de cette mémoire est envoyé tel quel vers le fichier précédant d'une adresse de 7 octets que voici :

FE Indicateur de fichier binaire

ee : Adresse en 16 bits inversée où stocker le fichier lors de son chargement en mémoire.

yy : Adresse en 16 bits inversée où stocker le dernier octet du fichier lors de son chargement en mémoire

ee : Adresse en 16 bits inversée où démarrer l'exécution du programme
 ea : lorsque le chargement en mémoire se fait avec l'option 'R',

?? Ici commence le programme ou l'image.

4.8.4 Les fichiers séquentiels.

Les fichiers séquentiels n'ont pas d'indrétil de type de fichier, mais, par contre, sont terminés par un marque de fin de fichier (code 1A). Ils sont créés par l'instruction Basic "OPEN" et remplis par l'instruction "PRINT#".

A ce sujet, il est bon de rappeler que l'instruction "PRINT#", insère automatiquement une marque de séparation entre les données numériques (code espace), mais pas entre les chaînes de caractères et qu'elle ajoute CR LF (code 0D-0A) à la fin de chaque ligne. Elle ne se termine pas par le code point-virgule. Toutes les données sont écrites en ASCII. Ainsi, le portion de programme Basic suivant créera le fichier F1-dessous.

```
10 OPEN"TEST,BE" FOR OUTPUT AS #1
20 A="22A1A"SWISS"
30 PRINT#1,32,47A
40 PRINT#1,"FRANCE","BELGIQUE","J"
50 PRINT#1,"CANADA"
60 CLOSE#1
```

32-47 225 01FRANCEBELGIQUESWISSCANADA

Le double caractère repris 16 le paire de caractères CR-LF (code 0D-0A), le caractère # représente le code de fin de fichier (1A).

Lorsque ce fichier sera lu par l'oreille de programme que voici,

200 IN#1,A,B,C,D,A1,B1,C1,D1

```
A contiendra 32.
B contiendra 47.
C contiendra 225.
A1 contiendra FRANCEBELGIQUESWISS.
B1 contiendra CANADA.
Ce et D1 ne contiendront rien.
```

En fait, comme dans tout INOUT, le virgule sert de séparateur des données (lignes). Comme l'instruction PRINT# n'insère pas la fin de ligne pour les chaînes de caractères, le variable A1 reçoit FRANCEBELGIQUESWISS en un seul mot. Pour éviter ce problème, il faut rajouter l'instruction de la ligne 40 en plaçant une virgule entre les chaînes.

```
40 PRINT#1,"FRANCE","BELGIQUE","J"
50 PRINT#1,"CANADA"
```

Donc, n'oubliez pas que seul la virgule, le CR LF (code 0D-0A) peuvent servir de séparateur de données de type chaîne.

4.8.5 Low tide/low to acute drought.

Le fichier à accès direct ne contient pas non plus d'histogramme de type de fichier ni de marque de fin de fichier. Les données sont sauvees en ASCII sauf pour les constantes et variables numériques qui sont sauvees telles qu'elles se trouvent en memoire. C'est à dire sous le format 2 octets pour les valeurs entieres, 4 octets pour les valeurs simples precision et 8 octets pour les valeurs doubles precision. Voici ce que l'on obtient avec le petit programme suivant.

```

10 OPEN "FICHER.DIR" AS #1
20 FIELD#1,6 AS Z1t,2 AS Z2t,t AS Z3t,8 AS Z4t
30 A$="DB1C":A2=1630:tA1=12345678901234
40 LSET Z1$=A2
50 LSET Z2$=MKC(tA1)
60 LSET Z3$=MKC(tA1)
70 LSET Z4$=MKC(tA1)
80 PUT t1,1
90 CLOSE #1:END

```

```

12 0
13 1
14 2
15 3
16 4
17 5
18 6
19 7
20 8
21 9
22 A
23 B
24 C
25 D
26 E
27 F
28 G
29 H
30 I
31 J
32 K
33 L
34 M
35 N
36 O
37 P
38 Q
39 R
40 S
41 T
42 U
43 V
44 W
45 X
46 Y
47 Z
48 [
49 \
50 ]
51 ^
52 _
53 `
54 a
55 b
56 c
57 d
58 e
59 f
60 g
61 h
62 i
63 j
64 k
65 l
66 m
67 n
68 o
69 p
70 q
71 r
72 s
73 t
74 u
75 v
76 w
77 x
78 y
79 z
80 {
81 |
82 }
83 ~
84 !
85 "
86 #
87 $
88 %
89 &
90 '
91 (
92 )
93 *
94 +
95 ,
96 -
97 .
98 /
99 :
100 ;
101 <
102 =
103 >
104 ?
105 @
106 [
107 \
108 ]
109 ^
110 _
111 `
112 a
113 b
114 c
115 d
116 e
117 f
118 g
119 h
120 i
121 j
122 k
123 l
124 m
125 n
126 o
127 p
128 q
129 r
130 s
131 t
132 u
133 v
134 w
135 x
136 y
137 z
138 {
139 |
140 }
141 ~
142 !
143 "
144 #
145 $
146 %
147 &
148 '
149 (
150 )
151 *
152 +
153 ,
154 -
155 .
156 /
157 :
158 ;
159 <
160 =
161 >
162 ?
163 @
164 [
165 \
166 ]
167 ^
168 _
169 `
170 a
171 b
172 c
173 d
174 e
175 f
176 g
177 h
178 i
179 j
180 k
181 l
182 m
183 n
184 o
185 p
186 q
187 r
188 s
189 t
190 u
191 v
192 w
193 x
194 y
195 z
196 {
197 |
198 }
199 ~
200 !
201 "
202 #
203 $
204 %
205 &
206 '
207 (
208 )
209 *
210 +
211 ,
212 -
213 .
214 /
215 :
216 ;
217 <
218 =
219 >
220 ?
221 @
222 [
223 \
224 ]
225 ^
226 _
227 `
228 a
229 b
230 c
231 d
232 e
233 f
234 g
235 h
236 i
237 j
238 k
239 l
240 m
241 n
242 o
243 p
244 q
245 r
246 s
247 t
248 u
249 v
250 w
251 x
252 y
253 z
254 {
255 |
256 }
257 ~
258 !
259 "
260 #
261 $
262 %
263 &
264 '
265 (
266 )
267 *
268 +
269 ,
270 -
271 .
272 /
273 :
274 ;
275 <
276 =
277 >
278 ?
279 @
280 [
281 \
282 ]
283 ^
284 _
285 `
286 a
287 b
288 c
289 d
290 e
291 f
292 g
293 h
294 i
295 j
296 k
297 l
298 m
299 n
300 o
301 p
302 q
303 r
304 s
305 t
306 u
307 v
308 w
309 x
310 y
311 z
312 {
313 |
314 }
315 ~
316 !
317 "
318 #
319 $
320 %
321 &
322 '
323 (
324 )
325 *
326 +
327 ,
328 -
329 .
330 /
331 :
332 ;
333 <
334 =
335 >
336 ?
337 @
338 [
339 \
340 ]
341 ^
342 _
343 `
344 a
345 b
346 c
347 d
348 e
349 f
350 g
351 h
352 i
353 j
354 k
355 l
356 m
357 n
358 o
359 p
360 q
361 r
362 s
363 t
364 u
365 v
366 w
367 x
368 y
369 z
370 {
371 |
372 }
373 ~
374 !
375 "
376 #
377 $
378 %
379 &
380 '
381 (
382 )
383 *
384 +
385 ,
386 -
387 .
388 /
389 :
390 ;
391 <
392 =
393 >
394 ?
395 @
396 [
397 \
398 ]
399 ^
400 _
401 `
402 a
403 b
404 c
405 d
406 e
407 f
408 g
409 h
410 i
411 j
412 k
413 l
414 m
415 n
416 o
417 p
418 q
419 r
420 s
421 t
422 u
423 v
424 w
425 x
426 y
427 z
428 {
429 |
430 }
431 ~
432 !
433 "
434 #
435 $
436 %
437 &
438 '
439 (
440 )
441 *
442 +
443 ,
444 -
445 .
446 /
447 :
448 ;
449 <
450 =
451 >
452 ?
453 @
454 [
455 \
456 ]
457 ^
458 _
459 `
460 a
461 b
462 c
463 d
464 e
465 f
466 g
467 h
468 i
469 j
470 k
471 l
472 m
473 n
474 o
475 p
476 q
477 r
478 s
479 t
480 u
481 v
482 w
483 x
484 y
485 z
486 {
487 |
488 }
489 ~
490 !
491 "
492 #
493 $
494 %
495 &
496 '
497 (
498 )
499 *
500 +
501 ,
502 -
503 .
504 /
505 :
506 ;
507 <
508 =
509 >
510 ?
511 @
512 [
513 \
514 ]
515 ^
516 _
517 `
518 a
519 b
520 c
521 d
522 e
523 f
524 g
525 h
526 i
527 j
528 k
529 l
530 m
531 n
532 o
533 p
534 q
535 r
536 s
537 t
538 u
539 v
540 w
541 x
542 y
543 z
544 {
545 |
546 }
547 ~
548 !
549 "
550 #
551 $
552 %
553 &
554 '
555 (
556 )
557 *
558 +
559 ,
560 -
561 .
562 /
563 :
564 ;
565 <
566 =
567 >
568 ?
569 @
570 [
571 \
572 ]
573 ^
574 _
575 `
576 a
577 b
578 c
579 d
580 e
581 f
582 g
583 h
584 i
585 j
586 k
587 l
588 m
589 n
590 o
591 p
592 q
593 r
594 s
595 t
596 u
597 v
598 w
599 x
600 y
601 z
602 {
603 |
604 }
605 ~
606 !
607 "
608 #
609 $
610 %
611 &
612 '
613 (
614 )
615 *
616 +
617 ,
618 -
619 .
620 /
621 :
622 ;
623 <
624 =
625 >
626 ?
627 @
628 [
629 \
630 ]
631 ^
632 _
633 `
634 a
635 b
636 c
637 d
638 e
639 f
640 g
641 h
642 i
643 j
644 k
645 l
646 m
647 n
648 o
649 p
650 q
651 r
652 s
653 t
654 u
655 v
656 w
657 x
658 y
659 z
660 {
661 |
662 }
663 ~
664 !
665 "
666 #
667 $
668 %
669 &
670 '
671 (
672 )
673 *
674 +
675 ,
676 -
677 .
678 /
679 :
680 ;
681 <
682 =
683 >
684 ?
685 @
686 [
687 \
688 ]
689 ^
690 _
691 `
692 a
693 b
694 c
695 d
696 e
697 f
698 g
699 h
700 i
701 j
702 k
703 l
704 m
705 n
706 o
707 p
708 q
709 r
710 s
711 t
712 u
713 v
714 w
715 x
716 y
717 z
718 {
719 |
720 }
721 ~
722 !
723 "
724 #
725 $
726 %
727 &
728 '
729 (
730 )
731 *
732 +
733 ,
734 -
735 .
736 /
737 :
738 ;
739 <
740 =
741 >
742 ?
743 @
744 [
745 \
746 ]
747 ^
748 _
749 `
750 a
751 b
752 c
753 d
754 e
755 f
756 g
757 h
758 i
759 j
760 k
761 l
762 m
763 n
764 o
765 p
766 q
767 r
768 s
769 t
770 u
771 v
772 w
773 x
774 y
775 z
776 {
777 |
778 }
779 ~
780 !
781 "
782 #
783 $
784 %
785 &
786 '
787 (
788 )
789 *
790 +
791 ,
792 -
793 .
794 /
795 :
796 ;
797 <
798 =
799 >
800 ?
801 @
802 [
803 \
804 ]
805 ^
806 _
807 `
808 a
809 b
810 c
811 d
812 e
813 f
814 g
815 h
816 i
817 j
818 k
819 l
820 m
821 n
822 o
823 p
824 q
825 r
826 s
827 t
828 u
829 v
830
```

4.8.6 Leaf-fishers NBX006

Il existe deux types particuliers de fichier MSDOS : les fichiers .COM et .BAT. Il n'est aucun des deux un format spécifique, le fichier .COM est simplement le texte de code qui forme un programme exécutable. Le fichier .BAT est un fichier ASCII donc traité par le code ASCII contenant une liste de commandes du MSDOS séparées par CR-LF.

Certains assembleurs comme MACROS8 de MICROSOFT ou certains compilateurs Pascal, Fortran, Cobol ou C peuvent générer des fichiers ayant une structure particulière propre, les .REL, .CRF, .HEX pour MACROS8. Reportez-vous à la documentation de ces divers assembleurs pour plus de renseignements.

4.9 Le formatage physique d'une piste

Peppione pour les non-initiés que le mariage d'une
diaquette vierge est nécessaire avant toute utilisation de
celle-ci...

L'opération de forage est celle par laquelle le hardware, piloté par un programme spiral, va découper chaque piste de la disquette en 8 ou 9 secteurs suivant le format choisi.

Potentiellement, une piste vierge est capable d'accueillir 4250 octets. Cependant, en MEX comme dans tout autre système d'ailleurs, l'utilisateur n'aurt en fait droit qu'à 9 secteurs de 512 octets soit 4608 octets par piste. On son donc passé les 4612 autres octets?

Ille vult tout diaphanum servare et erit ut 'enclavement' pour les 9 secteurs. Cet enclavement sert pour attice s'annoncer le numéro du secteur qui va bientôt passer sous le titre de lecture et de pressoir la vibration de paramètres comme le nombre de points et de titre.

Il va aussi servir à limiter un certain nombre de valeurs dans le but de tolérer certaines variations de la vitesse du moteur ou des caractéristiques physiques de la diquette.

Voici le format utilisé pour les diquettes de 90°

Le 1er octet est le numéro de la diquette (1 à 10), par défaut sont des diquettes hardware pour le 1er, nous prévoyons qu'il y ait une distinction par moteur.

7 Y Description	8 Y Description	9 Y Description	10 Y Description	11 Y Description	12 Y Description	13 Y Description	14 Y Description	15 Y Description	16 Y Description	17 Y Description	18 Y Description	19 Y Description	20 Y Description	21 Y Description	22 Y Description	23 Y Description	24 Y Description	25 Y Description	26 Y Description	27 Y Description	28 Y Description	29 Y Description	30 Y Description	31 Y Description	32 Y Description	33 Y Description	34 Y Description	35 Y Description	36 Y Description	37 Y Description	38 Y Description	39 Y Description	40 Y Description	41 Y Description	42 Y Description	43 Y Description	44 Y Description	45 Y Description	46 Y Description	47 Y Description	48 Y Description	49 Y Description	50 Y Description	51 Y Description	52 Y Description	53 Y Description	54 Y Description	55 Y Description	56 Y Description	57 Y Description	58 Y Description	59 Y Description	60 Y Description	61 Y Description	62 Y Description	63 Y Description	64 Y Description	65 Y Description	66 Y Description	67 Y Description	68 Y Description	69 Y Description	70 Y Description	71 Y Description	72 Y Description	73 Y Description	74 Y Description	75 Y Description	76 Y Description	77 Y Description	78 Y Description	79 Y Description	80 Y Description	81 Y Description	82 Y Description	83 Y Description	84 Y Description	85 Y Description	86 Y Description	87 Y Description	88 Y Description	89 Y Description	90 Y Description	91 Y Description	92 Y Description	93 Y Description	94 Y Description	95 Y Description	96 Y Description	97 Y Description	98 Y Description	99 Y Description	100 Y Description						
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100

```

10 4E Zone de l'index. Index gapi
12 00 Synchronisation du PIG. (Phase Lock Oscillator)
3 3F Marqueur d'index. Hardware Sector mark
1 1F Identificateur d'index. Index id.1
26 4E Zone de la tête d'index. (Head Index gapi)
12 00 Synchronisation de PLO. (Phase Lock Oscillator)
3 3F Marqueur de secteur. Hardware Sector mark
1 1F Identificateur de tête de secteur. (Header id.1)
1 2E Numéro de la piste. (Track number)
1 2E Numéro de la tête. (Head number)
1 3E Numéro de secteur dans la piste. (Physical Sector Number)
2 2E Caractères de redondance cyclique. (CRC)
2 4E Zone tête-secteur. (Sector gapi)
12 00 Synchronisation du PLO.
3 3F Marqueur de secteur. Hardware Sector mark
1 1F Identificateur de données. (Data id.1)
12 2E Zone des données de secteur. (Data)
2 2E Caractères de redondance cyclique. (CRC)
3 4E Zone inter-secteurs. (Inter sector gapi)
450 4E Zone d'avant-index. (File) gap

```

La carte mémoire de la zone RAM utilisée par le MSX-DOS
et le Disk-Basic

Lorsque vous faites l'acquisition d'une première unité de disquette, vous recevez en même temps son interface. Cet interface contient tout le hardware nécessaire au pilotage du disque y compris une ROM de 16K contenant :

- 1) Le Disk Basic programmé par MICROSOFT
- 2) Le Kernel MSXDOS (Noyau MSXDOS) programmé par MICROSOFT
- 3) Le Disk Driver (Piloté du disque) programmé par le fabricant du hardware de cet interface

Le fait que le Disk Driver soit programmé par le fabricant du Hardware autorise plus de souplesse dans la conception du Hardware. Ainsi certaines disquettes pourraient avoir des secteurs de 128, 256, 512 ou 1024 octets, ou encore être des disques durs de grande capacité ou même des disques à Laser.

Malgré d'autre part, certains fabricants ne respectent pas toujours la lettre la norme imposée par MICROSOFT d'où certaines différences de fonctionnement ou même certaines erreurs.

Avant de parler de la carte mémoire proprement dite, il faut citer et expliquer les 3 méthodes d'alimentation d'un MSX avec disquette(s), car elles influencent la taille et la configuration de la mémoire Ram réservée.

1) Allumer l'ordinateur en laissant le doigt sur la touche Majuscule jusqu'à un bip sonore provoque que le Disk-Basic n'est pas installé. Vous aurez ainsi un MSX avec 28015 octets libres mais pas d'accès au(x) disque(s). Cette manipulation est parfois nécessaire avec certaines cassettes ou cartouches de jeux Eddy2, par exemple lorsque la mémoire libre est insuffisante avec le Disk-Basic installé.

2) Allumer l'ordinateur normalement fait en sorte que le système réserve de l'espace en mémoire pour deux unités de disquette par interface délecté, qu'il y ait un ou deux lecteurs connectés à chacun de ces interfaces.

3) Allumer l'ordinateur en laissant le doigt sur la touche CTRL (Control) jusqu'à un bip sonore provoque que le système ne réserve de l'espace en mémoire que pour le premier lecteur de chaque interface. Si le second lecteur de certains de ces interfaces était allumé durant le démarrage de l'ordinateur, alors de l'espace en mémoire est également réservé pour ces unités allumées. Cette technique permet d'ajuster la mémoire réservée au minimum requis pour la configuration dont vous avez besoin. Première différence entre fabricants, la version MSX2 V8B235 de PHILIPS ne rallie que la première unité de chaque interface même si la

seconde état alluée.

Une autre liberté a été prise par JVC (Version JVC72) vis à vis de la norme de MICROSOFT et cela a beaucoup plus de conséquence. En effet, la version d'interface JVC72 ne réserve de l'espace pour la FAT que pour des disquettes de 360k. Cela a pour avantage de laisser 24456 octets libres au lieu de 23432, mais par contre provoque que vous ne pourrez pas y connecter un lecteur de 720K ou en tous cas vous ne pourrez en accéder que les 680 premiers K.

5.1 Le cart. mémoire générale

5.1.1 Taille des zones mémoire pour le DOS et le Disk-Basic

Vous savez certainement que le Basic normal de votre MSX s'est déjà réservé la portion de mémoire Ram qui s'étend de Hexa F380 à FFFF. Cette région de communication contient une série de paramètres, de lampons et de constantes que LE LIVRE DU MSX de Daniel MARTIN vous expliquera. Je ne peux que vous en conseiller vivement la lecture.

La région de communication réservée pour le DOS est constituée de deux parties. La première est fixe et s'étend de la position hexa F1C9 à F37F, c-à-d. juste sous la région de communication du Basic. La seconde partie est de longueur variable et dépend en fait du nombre et du type de disques reconnus par votre MSX lors de l'allumage et du fabricant du contrôleur dont vous êtes équipés.

A titre d'exemple, cette seconde partie de la région de communication disque peut s'étendre de F60F à F1CB pour un MSX à un seul contrôleur (Version JVC72) un seul lecteur 360k : Touche CTRL enfoncée durant l'allumage, et de 5E79 à F1CB pour un MSX à un contrôleur à un ou deux lecteurs de 720k démarrage normal.

L'occupation mémoire maximale étant de 698D à F1CB pour un MSX de huit unités de 720K : Version d'interface Compact démarrage normal. Cette configuration ne laisse que 9344 octets libres pour le Basic et très peu de billets dans votre poche (quill).

5.1.2 Initialisation du système MSX avec disquettes

La procédure d'initialisation des MSX est la suivante : L'ordinateur va d'abord rechercher si une ROM est présente en 4000H ou en 8000H en balayant chaque slot à partir du slot primaire 0 jusqu'au slot primaire 3. Si le slot primaire balayé est un slot étendu, il va aussi balayer les quatre slots secondaires de ce slot primaire.

Si le système découvre une ROM identifiable par la présence des lettres AB en deux premières positions de cette ROM, il exécutera la routine d'initialisation de cette ROM dont l'adresse se trouve dans les deux positions qui suivent les lettres AB.

Si la ROM détectée est une ROM de Contrôleur disque, la routine d'initialisation va réserver 439 positions RAM en dessous de la région de communication du Basic (F380). Ces 439 positions constituent la région de communication fixe des disques (F1C9 à F37F).

Enfin cette routine va réserver en dessous des 439 octets, 8 octets pour ce contrôleur disque qui est appelé le Contrôleur 0 puisque'il est le premier détecté. Ces 8 octets

5.1.4 Différences entre les occupations mémoire MSX-DOS et Disk-Basic.

Les deux premiers tableaux représentent la carte mémoire commune au DISK-BASIC et au MSXDOS. Le reste de la réservation mémoire va dépendre du choix que vous ferez de travailler en Disk-Basic ou en MSXDOS.

Au cas où la disquette, insérée dans le disque à la lors de l'allumage de l'ordinateur, ne contient pas le fichier MSXDOS.SYS, la système ne vous laissera pas le choix et vous imposera de travailler en Disk-Basic.

Si votre disquette contient le fichier MSXDOS.SYS et le fichier COMMAND.COM, ce sera le MSXDOS qui sera activé et le message "MSX-DOS Version 1.03.1" apparaîtra suivi de l'indicateur du mode commande A; vous pourrez alors choisir d'appeler un programme MSXDOS ou taper une commande comme DIR, DATE, TYPE etc..., ou encore la commande BASIC qui vous renverra vers le Disk-Basic.

Voici d'abord le tableau représentant la partie de la carte mémoire. Ce tableau se situe juste au dessous du dernier tableau FAT.

DISK-BASIC	MSX-DOS
Zone de 7 x 37 octets réservée pour les 7 File Control Blocks (FCB0 à FCB6)	Zone de 648 octets (JVCKT2=645) contenant les routines de communication des slots.
Routine en langage machine utilisée par les instructions BLOAD et BSAVE	Routine destinée aux fonctions du MSXDOS (Fonctions 00H à 30H)
Zone que vous pouvez réserver pour vos programmes en langage machine par l'instruction CLEAR et qui s'étend vers la bas	Zone de pile utilisateur (stack) par défaut à l'appel d'un programme MSXDOS, elle peut être recouverte par le programme utilisateur
Zone des tampons de fichiers. Le nombre de tampons dépend de MAXFILES. Le tampon réservé au fichier 0 est le plus bas. Cette zone s'agrandit vers la bas.	Routine des commandes du MSXDOS, DIR, TYPE, COPY, DEL, etc... Cette zone peut être recouverte par votre programme si nécessaire.
Zone réservée au stockage de vos variables "chaîne" de 200 octets par défaut mais ajustable par CLEAR et qui s'étend vers la bas.	Z O N E
Zone de la pile Basic (Stack) qui s'étend vers la bas.	
Zone libre. D'une certaine zone sera nulle, OUT OF MEMORY sera affichée.	R E S E R V E E
Zone des variables dimensionnées qui s'étend vers le haut.	A U P R O G R A M M E
Zone des variables simples qui s'étend vers le haut.	
Zone du texte du programme BASIC qui s'étend de 8000H vers le haut.	U T I L I S A T E U R
ROM de 16 K contenant le BASIC de l'adr. 4000H à 7FFFH	
ROM de 16 K contenant le BIOS MSX de l'adresse 0000H à 3FFFH.	Page 0 du MSXDOS 10000H - 01000H

Il s'agit des routines qui permettent de lire les 42 fonctions offertes par le MSXDOS et décrites dans le chapitre 9. Cette zone est le seul endroit où l'on peut accéder à la ROM de contrôle des disques qui contient les routines propres à la ROM de lecture des disques.

ZONE DE LA PILE (STACK)

C'est l'emplacement de la pile (stack) réservée au programme qui tourne sous MSXDOS. On lui a réservé une taille de 256 octets. Si vous voulez utiliser les commandes de MSXDOS à l'intérieur d'un autre programme, il vous faut faire un usage important de la pile. Il s'agit d'attribuer de l'espace à la pile sous la zone des commandes du MSXDOS afin de ne pas traverser la durée de la zone par une pile qui s'étendrait trop vers le bas.

COMMANDES DU MSIDOS:

C'est la zone où se trouvent les commandes du MSIDOS (DIR, TYPE, DATE, ...). Cette zone occupe 1024 octets sans la version 1.11 de COMMAND.COM qui est susceptible de changer de longueur dans d'autres versions. Elle a l'avantage de pouvoir être restaurée par votre programmeur en effet, elle se recharge automatiquement dès le tir de votre programme.

ZONE DU PROGRAMME:

C'est à cet endroit que s'implantent vos programmes MSXDOS en langage machine. Cette zone commence à l'adresse 0100 et s'étend jusqu'à la fin des fonctions du MSXDOS et sous une autre version de la zone (stack) interne à votre programme. Ainsi, elle est limitée par la fin de la pile ou par la fin des commandes du MSIDOS.

Pour vous donner une idée de la taille de cette zone, nous supposons que vous avez un seul contrôleur disque et que vous avez écrit le système normalement. Dans ce cas, vous disposez de 54,221 octets. Juste un peu plus de la moitié de la zone des commandes du MSIDOS.

Sur un MSX2 avec 128 K de RAM utilisateur (soit la V08235 de PHILIPS), il y a 63,536 octets supplémentaires, mais uniquement par l'ajout de la mémoire MAPPER décrite au chapitre 9, ce qui leur offre 119,757 octets réservés à leur programme, ce qui n'est pas mal du tout pour un ordinateur familial à base de 180°.

MSIDOS PART 01

Les 256 premières caractères de RAM à l'adresse 0 sont réservés pour le MSXDOS. Ils sont décrits plus loin. Sachez qu'on y trouve des points d'entrée, les FCB par défaut et le DATA IDIOM Transfer Address par défaut.

5.2 Table des commandes des unités de disquettes

Cette table est implantée dans la région de manipulation du BIOS à l'adresse F21H de la sorte que vous pouvez un programme chargé depuis une cassette (sans avoir le système) à l'essai et tester les commandes de disque ou non.

La table est constituée de deux octets par commande soit huit octets. Le premier octet de la paire indique le nombre 10, ou 21 de disques logiques (et non pas physiques) connectés au contrôleur. Le deuxième octet donne la parité de la commande de la ROM. Le bit 0 est toujours 00. Le contrôleur sous la table:

E 0 0 0 0 0 0 0 0 est le numéro de la première commande.
 SS est le numéro de la commande secondaire 10.
 E=11, E=11 est le bit principal est toujours en
 secondaires.
 E=0 est le bit principal n'est pas toujours.

1021: Nombre de disques logiques du Contrôleur 0
 1022: Bit du Contrôleur 0
 1023: Nombre de disques logiques du Contrôleur 1
 1024: Bit du Contrôleur 1
 1025: Nombre de disques logiques du Contrôleur 2
 1026: Bit du Contrôleur 2
 1027: Nombre de disques logiques du Contrôleur 3
 1028: Bit du Contrôleur 3

L'intérêt de cette table est d'abord qu'elle permet de connaître le nombre de disques implantés par Contrôleur sans avoir à vous indiquer sans quel bit et se trouve un contrôleur donné.

Vous verrez dans le chapitre 4 qu'il est nécessaire de connaître le bit et vous voulez connaître les points d'entrée des ROM des Contrôleurs disques. De plus, la ROM indique le bit du Contrôleur est directement associable par les routines de manipulation de disques.

5.3. Table des Zones de Travail réservées par la ROM Disk

Cette table est ainsi implantée dans le région de communication du BASIC de l'adresse FD09 à FD6B et a donc une longueur de 128 octets. Elle est constituée de deux octets par ROM qu'il est possible d'installer dans un MSX.

Comme la norme MSX prévoit que la mémoire peut être partagée en 4 slots primaires, eux-mêmes partagés en 4 slots secondaires et contenant chacun 4 blocs de 16k, nous arrivons au total de 64 ROM possibles d'où cette longueur de table de 128 octets.

Les deux octets réservés par ROM sont garantis par la norme MSX comme étant à l'usage exclusif de cette ROM.

Pour trouver l'adresse des deux octets spécifiquement réservés à un Contrôleur disque, il suffit d'employer la formule suivante:

$$\text{Adresse} = \text{FD09} + \{ 32 \times \text{SP} \} + \{ 8 \times \text{SS} \} + \{ 2 \times \text{BK} \}$$

où

SP est le numéro du slot primaire;

SS est le numéro du slot secondaire;

BK est le numéro du Bank (0 = 0000H 1 = 4000H 2 = 8000H 3 = C000H)

Dans le cas d'un Contrôleur disque le numéro du Bank est toujours 1 soit 4000H. A cette adresse se trouvent donc les deux octets exclusivement réservés au Contrôleur disque présent dans le slot secondaire SS du slot primaire BP.

Dans notre cas ces deux octets forment un pointeur d'une zone mémoire RAM où se trouve la Controller Work Area (zone de travail du contrôleur). Le contenu de la CWA dépend du fabricant de l'interface et sera décrit plus loin dans ce chapitre.

5.4 Le Disk Parameter Block

Il y a autant de DPB (Bloc de Paramètres du Disque) qu'il y a de disques reconnus par le système.

Un DPB n'est pas nécessairement attaché à un disque physique, ainsi lorsque vous n'avez qu'un seul disque physique, vous disposez malgré tout de deux disques logiques appelés A: et B: (voir chapitre 2) et donc il y a deux DPB.

L'adresse d'un DPB en mémoire est donnée par son pointeur qui se trouve dans la table des DPB à l'adresse F335. Pour trouver l'adresse du DPB0 (Disque A:), il suffit de lire les deux octets présents dans TABLE + (2 x No disque) soit $F335 + (2 \times 0) = F335$.

Si vous cherchez le DPB3 (Disque D:), lisez les deux octets présents à l'adresse $F335 + (2 \times 3) = F33B$. D'autre part, comme il n'y a jamais qu'un seul disque activé à la fois, il est possible de trouver le DPB du dernier disque activé en prenant le contenu du pointeur F243 (DPB en activité).

Un DPB est constitué de 21 octets donnant des informations sur les caractéristiques physiques ou structurales de son disque. Voici sa description:

Positi	Nom	Description de la zone
0	DE	Drive number: 0 à 7 = F: H aucun est attribué en CDS.
1	MEDIF	Type de disque encode comme suit: 11111111 10 : 80 pistes 00 : 16 sect./piste 10 : 80 pistes 01 : 80 pistes 01 : 80 pistes 10 : 80 pistes 01 : 80 pistes 11 : 80 pistes
2-2	REC211	Longueur physique du secteur en octets.
3	DIRECT	Directory Name: Nom qui permet d'identifier la position d'un libéro dans un secteur Directory d'après la position globale
5	DIRECT	Directory Offset: Possibilité de 2 indicateurs 1 le nombre de libéros dans un secteur Directory = 2 - DIRECT + Nombre
6	CLUSTER	CLUSTER Name: Mapping entre le cluster et le secteur secteur à un cluster qui est utilisé. Value CLUSTER = 11
7	CLUSTER	Indique le nombre de secteurs d'un cluster.
9	PIRPI	PIRPI bit vector: PIRPI bit inversé le bit du premier secteur le 1 est sur disque.
10	PIRPI	PIRPI Count: nombre de PIRPI de disque
11	MAXENT	Maximum Entry number within the libéro que l'on peut placer dans le Directory
12-13	PIRPI	PIRPI Vector: Indique si le bit est inversé le bit du premier secteur du premier PIRPI
14-15	MAXENT	Maximum Entry: donne le bit inversé le bit de l'unique utilisateur unique du disque en nombre de libéros.
16	PIRPI	PIRPI bit vector: PIRPI bit inversé le bit du premier secteur le 1 est sur disque.
17-18	PIRPI	PIRPI Directory: donne le bit inversé le bit de l'unique utilisateur unique du disque en nombre de libéros.
19-20	PIRPI	PIRPI Directory: donne le bit inversé le bit de l'unique utilisateur unique du disque en nombre de libéros.

Il y a eu autant de CNN, donc de travail pour Contilieu
siempre que d'interface pour disque. La présence de celle
sont n'est pas rendue obligatoire par la norme, mais est
un détail l'apparition du fabricant Contilieu et
du Strat Driver 1-3-3, le programme de piratage du hardware
de disque. Cependant, dans tous les Disk Drivers que j'ai
pu analyser, celle-ci n'est pas toujours présente, avec une
lecture de son octet.

*Pour plus d'infos : www.cirad.fr

[illegible]

1 TDO
2 ligne D16 de D : D'ensemble il s'agit d'un physicien D.
3 Ce schéma est décomposé pour D16 P Phénom
4 question sur la décomposition, ligne D'ensemble
5 dans la direction, il s'agit d'un physicien P D
6 le système chargé d'élaborer les données de la
7 loi et de la loi, au moment de la loi de la loi
8 indiquée dans la loi de la loi, la loi de la loi
9 l'ensemble de la loi de la loi de la loi de la loi
10 Cette décomposition est décomposée en la loi de la loi
11 décomposée en la loi de la loi de la loi de la loi
12 Si la loi de la loi de la loi de la loi, le système est
13 que dans la loi de la loi de la loi de la loi de la loi
14 donc de la loi de la loi de la loi de la loi de la loi

2 101 1 Year Day 1 : name of driver part of day 1 101

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016 1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1

```

1  * TKO : Track disk 0 : Numerique sur pucele piole le
2  *      : etape physique 0 se trouve pueul l'en travaille
3  *      : avec la etape physique 1

```

5 TKI Traje dyak I : Conno ci-dessus pour la liqee I

```

0 34D : deleted Logical Disk : dernier disque logique
1      : sélectionné. Cette position permet de savoir :
2      : quel disque logique est présent dans le disque
3      : présente dans le disque physique dans les
4      : évènements d'un disque déviant.

```

* 2 : OCT = Disk Count : nombre de disques physiques de lecture

3.6 Le FCB Control Block

Le FCB Control Block (FCB) de contrôle du fichier est un dispositif spécial dans le langage des fichiers par le langage machine.

C'est une zone de 37 octets qui contient toutes les informations nécessaires à la manipulation correcte d'un fichier ouvert (celles que son nom, l'unité sur laquelle le fichier se trouve, son emplacement sur le disque, le quel aurait de l'histoire la prochaine lecture ou écriture doit se faire, etc.).

Il existe deux types de FCB supportés par le MS-DOS. Le FCB DOS a un état d'attente complet par lequel supporte une grande proportion de programmes du système de fichiers, bien entendu, que les programmes de fichiers spécifiques pour le MS-DOS.

Pour cette raison, nous examinerons d'abord le FCB employé en CP/M et par le MS-DOS. Le FCB utilisé en Disk Basic était identique et celui du MS-DOS, vous n'en trouverez pas de description séparée.

Lorsque qu'un programme en langage machine doit accéder à un fichier, il devra, comme un utilisateur, passer par trois phases :

- 1 - L'ouverture du fichier
- 2 - Le lecture ou l'écriture du fichier
- 3 - Le fermeture du fichier

De plus, c'est à l'ouverture que toutes les informations nécessaires au système seront enregistrées comme dans l'instruction suivante :

```
OPEN "0:ADRESSE.111" AH of LEN=120
```

Cette instruction permet qu'il soit ouvert le fichier ADRESSE.111 au l'adresse de la zone d'adresse direct, avec une taille d'emplacement de 120 octets sous le nom de fichier 1. Par la suite, les phases 2 et 3 c-à-d, la manipulation et la fermeture du fichier, se feront par référence au nom de fichier 1.

En CP/M ou en MS-DOS, les 3 phases se feront non pas par référence à un numéro de fichier mais par référence d'un FCB dont le programme en langage machine devra simplement indiquer l'adresse.

Les FCB CP/M et MS-DOS doivent donc être créés par le programme avant l'ouverture du fichier. L'appareil d'ouverture du fichier acceptera ensuite le FCB créé par le programmeur en y ajoutant des données liant le fichier de la Directory correspondant au fichier.

Il y a deux différences majeures entre CP/M et MS-DOS :

1) De CP/M, le FCB a une taille fixe de 120 octets. En MS-DOS, la taille du record est programmable de 1 à 45535 octets.

2) En CP/M, l'allocation d'espace sur le disque pour un fichier se fait par l'extension de l'écriture de 0, CC. Dans le cas du MS-DOS, un programme CP/M, c'est-à-dire, demande 120 Records (120 Records de 120 octets soit 84-1024). La notion d'extension de 0, CC est connue et l'on veut programmer en CP/M et se positionner à l'adresse de la zone d'écriture de plus de 10K. Par contre, en MS-DOS, la notion d'extension de 0, CC est connue et l'on veut programmer en CP/M et se positionner à l'adresse de la zone d'écriture de plus de 10K. Par contre, en MS-DOS, la notion d'extension de 0, CC est connue et l'on veut programmer en CP/M et se positionner à l'adresse de la zone d'écriture de plus de 10K.

3.6.3 LE FCB DU CP/M.

Le FCB du CP/M est constitué de 36 octets dont voici le découpage. Les 16 premiers octets plus l'octet 32 doivent être remplis par le programmeur avant de l'OPEN.

Pos. Octet	Nom	Description de la zone
0	DR	Drive : la programmation indique ici le no. du disque sur lequel se trouve le fichier.
1	FC	0 - C : disque dur ; 1 - D : 5 1/4 ; 2 - 5 1/4 ; 3 - 5 1/4 ; 4 - 5 1/4 ; 5 - 5 1/4 ; 6 - 5 1/4 ; 7 - 5 1/4 ; 8 - 5 1/4 ; 9 - 5 1/4 ; 10 - 5 1/4 ; 11 - 5 1/4 ; 12 - 5 1/4 ; 13 - 5 1/4 ; 14 - 5 1/4 ; 15 - 5 1/4 ; 16 - 5 1/4 ; 17 - 5 1/4 ; 18 - 5 1/4 ; 19 - 5 1/4 ; 20 - 5 1/4 ; 21 - 5 1/4 ; 22 - 5 1/4 ; 23 - 5 1/4 ; 24 - 5 1/4 ; 25 - 5 1/4 ; 26 - 5 1/4 ; 27 - 5 1/4 ; 28 - 5 1/4 ; 29 - 5 1/4 ; 30 - 5 1/4 ; 31 - 5 1/4 ; 32 - 5 1/4 ; 33 - 5 1/4 ; 34 - 5 1/4 ; 35 - 5 1/4 ; 36 - 5 1/4 ; 37 - 5 1/4 ; 38 - 5 1/4 ; 39 - 5 1/4 ; 40 - 5 1/4 ; 41 - 5 1/4 ; 42 - 5 1/4 ; 43 - 5 1/4 ; 44 - 5 1/4 ; 45 - 5 1/4 ; 46 - 5 1/4 ; 47 - 5 1/4 ; 48 - 5 1/4 ; 49 - 5 1/4 ; 50 - 5 1/4 ; 51 - 5 1/4 ; 52 - 5 1/4 ; 53 - 5 1/4 ; 54 - 5 1/4 ; 55 - 5 1/4 ; 56 - 5 1/4 ; 57 - 5 1/4 ; 58 - 5 1/4 ; 59 - 5 1/4 ; 60 - 5 1/4 ; 61 - 5 1/4 ; 62 - 5 1/4 ; 63 - 5 1/4 ; 64 - 5 1/4 ; 65 - 5 1/4 ; 66 - 5 1/4 ; 67 - 5 1/4 ; 68 - 5 1/4 ; 69 - 5 1/4 ; 70 - 5 1/4 ; 71 - 5 1/4 ; 72 - 5 1/4 ; 73 - 5 1/4 ; 74 - 5 1/4 ; 75 - 5 1/4 ; 76 - 5 1/4 ; 77 - 5 1/4 ; 78 - 5 1/4 ; 79 - 5 1/4 ; 80 - 5 1/4 ; 81 - 5 1/4 ; 82 - 5 1/4 ; 83 - 5 1/4 ; 84 - 5 1/4 ; 85 - 5 1/4 ; 86 - 5 1/4 ; 87 - 5 1/4 ; 88 - 5 1/4 ; 89 - 5 1/4 ; 90 - 5 1/4 ; 91 - 5 1/4 ; 92 - 5 1/4 ; 93 - 5 1/4 ; 94 - 5 1/4 ; 95 - 5 1/4 ; 96 - 5 1/4 ; 97 - 5 1/4 ; 98 - 5 1/4 ; 99 - 5 1/4 ; 100 - 5 1/4 ; 101 - 5 1/4 ; 102 - 5 1/4 ; 103 - 5 1/4 ; 104 - 5 1/4 ; 105 - 5 1/4 ; 106 - 5 1/4 ; 107 - 5 1/4 ; 108 - 5 1/4 ; 109 - 5 1/4 ; 110 - 5 1/4 ; 111 - 5 1/4 ; 112 - 5 1/4 ; 113 - 5 1/4 ; 114 - 5 1/4 ; 115 - 5 1/4 ; 116 - 5 1/4 ; 117 - 5 1/4 ; 118 - 5 1/4 ; 119 - 5 1/4 ; 120 - 5 1/4 ; 121 - 5 1/4 ; 122 - 5 1/4 ; 123 - 5 1/4 ; 124 - 5 1/4 ; 125 - 5 1/4 ; 126 - 5 1/4 ; 127 - 5 1/4 ; 128 - 5 1/4 ; 129 - 5 1/4 ; 130 - 5 1/4 ; 131 - 5 1/4 ; 132 - 5 1/4 ; 133 - 5 1/4 ; 134 - 5 1/4 ; 135 - 5 1/4 ; 136 - 5 1/4 ; 137 - 5 1/4 ; 138 - 5 1/4 ; 139 - 5 1/4 ; 140 - 5 1/4 ; 141 - 5 1/4 ; 142 - 5 1/4 ; 143 - 5 1/4 ; 144 - 5 1/4 ; 145 - 5 1/4 ; 146 - 5 1/4 ; 147 - 5 1/4 ; 148 - 5 1/4 ; 149 - 5 1/4 ; 150 - 5 1/4 ; 151 - 5 1/4 ; 152 - 5 1/4 ; 153 - 5 1/4 ; 154 - 5 1/4 ; 155 - 5 1/4 ; 156 - 5 1/4 ; 157 - 5 1/4 ; 158 - 5 1/4 ; 159 - 5 1/4 ; 160 - 5 1/4 ; 161 - 5 1/4 ; 162 - 5 1/4 ; 163 - 5 1/4 ; 164 - 5 1/4 ; 165 - 5 1/4 ; 166 - 5 1/4 ; 167 - 5 1/4 ; 168 - 5 1/4 ; 169 - 5 1/4 ; 170 - 5 1/4 ; 171 - 5 1/4 ; 172 - 5 1/4 ; 173 - 5 1/4 ; 174 - 5 1/4 ; 175 - 5 1/4 ; 176 - 5 1/4 ; 177 - 5 1/4 ; 178 - 5 1/4 ; 179 - 5 1/4 ; 180 - 5 1/4 ; 181 - 5 1/4 ; 182 - 5 1/4 ; 183 - 5 1/4 ; 184 - 5 1/4 ; 185 - 5 1/4 ; 186 - 5 1/4 ; 187 - 5 1/4 ; 188 - 5 1/4 ; 189 - 5 1/4 ; 190 - 5 1/4 ; 191 - 5 1/4 ; 192 - 5 1/4 ; 193 - 5 1/4 ; 194 - 5 1/4 ; 195 - 5 1/4 ; 196 - 5 1/4 ; 197 - 5 1/4 ; 198 - 5 1/4 ; 199 - 5 1/4 ; 200 - 5 1/4 ; 201 - 5 1/4 ; 202 - 5 1/4 ; 203 - 5 1/4 ; 204 - 5 1/4 ; 205 - 5 1/4 ; 206 - 5 1/4 ; 207 - 5 1/4 ; 208 - 5 1/4 ; 209 - 5 1/4 ; 210 - 5 1/4 ; 211 - 5 1/4 ; 212 - 5 1/4 ; 213 - 5 1/4 ; 214 - 5 1/4 ; 215 - 5 1/4 ; 216 - 5 1/4 ; 217 - 5 1/4 ; 218 - 5 1/4 ; 219 - 5 1/4 ; 220 - 5 1/4 ; 221 - 5 1/4 ; 222 - 5 1/4 ; 223 - 5 1/4 ; 224 - 5 1/4 ; 225 - 5 1/4 ; 226 - 5 1/4 ; 227 - 5 1/4 ; 228 - 5 1/4 ; 229 - 5 1/4 ; 230 - 5 1/4 ; 231 - 5 1/4 ; 232 - 5 1/4 ; 233 - 5 1/4 ; 234 - 5 1/4 ; 235 - 5 1/4 ; 236 - 5 1/4 ; 237 - 5 1/4 ; 238 - 5 1/4 ; 239 - 5 1/4 ; 240 - 5 1/4 ; 241 - 5 1/4 ; 242 - 5 1/4 ; 243 - 5 1/4 ; 244 - 5 1/4 ; 245 - 5 1/4 ; 246 - 5 1/4 ; 247 - 5 1/4 ; 248 - 5 1/4 ; 249 - 5 1/4 ; 250 - 5 1/4 ; 251 - 5 1/4 ; 252 - 5 1/4 ; 253 - 5 1/4 ; 254 - 5 1/4 ; 255 - 5 1/4 ; 256 - 5 1/4 ; 257 - 5 1/4 ; 258 - 5 1/4 ; 259 - 5 1/4 ; 260 - 5 1/4 ; 261 - 5 1/4 ; 262 - 5 1/4 ; 263 - 5 1/4 ; 264 - 5 1/4 ; 265 - 5 1/4 ; 266 - 5 1/4 ; 267 - 5 1/4 ; 268 - 5 1/4 ; 269 - 5 1/4 ; 270 - 5 1/4 ; 271 - 5 1/4 ; 272 - 5 1/4 ; 273 - 5 1/4 ; 274 - 5 1/4 ; 275 - 5 1/4 ; 276 - 5 1/4 ; 277 - 5 1/4 ; 278 - 5 1/4 ; 279 - 5 1/4 ; 280 - 5 1/4 ; 281 - 5 1/4 ; 282 - 5 1/4 ; 283 - 5 1/4 ; 284 - 5 1/4 ; 285 - 5 1/4 ; 286 - 5 1/4 ; 287 - 5 1/4 ; 288 - 5 1/4 ; 289 - 5 1/4 ; 290 - 5 1/4 ; 291 - 5 1/4 ; 292 - 5 1/4 ; 293 - 5 1/4 ; 294 - 5 1/4 ; 295 - 5 1/4 ; 296 - 5 1/4 ; 297 - 5 1/4 ; 298 - 5 1/4 ; 299 - 5 1/4 ; 300 - 5 1/4 ; 301 - 5 1/4 ; 302 - 5 1/4 ; 303 - 5 1/4 ; 304 - 5 1/4 ; 305 - 5 1/4 ; 306 - 5 1/4 ; 307 - 5 1/4 ; 308 - 5 1/4 ; 309 - 5 1/4 ; 310 - 5 1/4 ; 311 - 5 1/4 ; 312 - 5 1/4 ; 313 - 5 1/4 ; 314 - 5 1/4 ; 315 - 5 1/4 ; 316 - 5 1/4 ; 317 - 5 1/4 ; 318 - 5 1/4 ; 319 - 5 1/4 ; 320 - 5 1/4 ; 321 - 5 1/4 ; 322 - 5 1/4 ; 323 - 5 1/4 ; 324 - 5 1/4 ; 325 - 5 1/4 ; 326 - 5 1/4 ; 327 - 5 1/4 ; 328 - 5 1/4 ; 329 - 5 1/4 ; 330 - 5 1/4 ; 331 - 5 1/4 ; 332 - 5 1/4 ; 333 - 5 1/4 ; 334 - 5 1/4 ; 335 - 5 1/4 ; 336 - 5 1/4 ; 337 - 5 1/4 ; 338 - 5 1/4 ; 339 - 5 1/4 ; 340 - 5 1/4 ; 341 - 5 1/4 ; 342 - 5 1/4 ; 343 - 5 1/4 ; 344 - 5 1/4 ; 345 - 5 1/4 ; 346 - 5 1/4 ; 347 - 5 1/4 ; 348 - 5 1/4 ; 349 - 5 1/4 ; 350 - 5 1/4 ; 351 - 5 1/4 ; 352 - 5 1/4 ; 353 - 5 1/4 ; 354 - 5 1/4 ; 355 - 5 1/4 ; 356 - 5 1/4 ; 357 - 5 1/4 ; 358 - 5 1/4 ; 359 - 5 1/4 ; 360 - 5 1/4 ; 361 - 5 1/4 ; 362 - 5 1/4 ; 363 - 5 1/4 ; 364 - 5 1/4 ; 365 - 5 1/4 ; 366 - 5 1/4 ; 367 - 5 1/4 ; 368 - 5 1/4 ; 369 - 5 1/4 ; 370 - 5 1/4 ; 371 - 5 1/4 ; 372 - 5 1/4 ; 373 - 5 1/4 ; 374 - 5 1/4 ; 375 - 5 1/4 ; 376 - 5 1/4 ; 377 - 5 1/4 ; 378 - 5 1/4 ; 379 - 5 1/4 ; 380 - 5 1/4 ; 381 - 5 1/4 ; 382 - 5 1/4 ; 383 - 5 1/4 ; 384 - 5 1/4 ; 385 - 5 1/4 ; 386 - 5 1/4 ; 387 - 5 1/4 ; 388 - 5 1/4 ; 389 - 5 1/4 ; 390 - 5 1/4 ; 391 - 5 1/4 ; 392 - 5 1/4 ; 393 - 5 1/4 ; 394 - 5 1/4 ; 395 - 5 1/4 ; 396 - 5 1/4 ; 397 - 5 1/4 ; 398 - 5 1/4 ; 399 - 5 1/4 ; 400 - 5 1/4 ; 401 - 5 1/4 ; 402 - 5 1/4 ; 403 - 5 1/4 ; 404 - 5 1/4 ; 405 - 5 1/4 ; 406 - 5 1/4 ; 407 - 5 1/4 ; 408 - 5 1/4 ; 409 - 5 1/4 ; 410 - 5 1/4 ; 411 - 5 1/4 ; 412 - 5 1/4 ; 413 - 5 1/4 ; 414 - 5 1/4 ; 415 - 5 1/4 ; 416 - 5 1/4 ; 417 - 5 1/4 ; 418 - 5 1/4 ; 419 - 5 1/4 ; 420 - 5 1/4 ; 421 - 5 1/4 ; 422 - 5 1/4 ; 423 - 5 1/4 ; 424 - 5 1/4 ; 425 - 5 1/4 ; 426 - 5 1/4 ; 427 - 5 1/4 ; 428 - 5 1/4 ; 429 - 5 1/4 ; 430 - 5 1/4 ; 431 - 5 1/4 ; 432 - 5 1/4 ; 433 - 5 1/4 ; 434 - 5 1/4 ; 435 - 5 1/4 ; 436 - 5 1/4 ; 437 - 5 1/4 ; 438 - 5 1/4 ; 439 - 5 1/4 ; 440 - 5 1/4 ; 441 - 5 1/4 ; 442 - 5 1/4 ; 443 - 5 1/4 ; 444 - 5 1/4 ; 445 - 5 1/4 ; 446 - 5 1/4 ; 447 - 5 1/4 ; 448 - 5 1/4 ; 449 - 5 1/4 ; 450 - 5 1/4 ; 451 - 5 1/4 ; 452 - 5 1/4 ; 453 - 5 1/4 ; 454 - 5 1/4 ; 455 - 5 1/4 ; 456 - 5 1/4 ; 457 - 5 1/4 ; 458 - 5 1/4 ; 459 - 5 1/4 ; 460 - 5 1/4 ; 461 - 5 1/4 ; 462 - 5 1/4 ; 463 - 5 1/4 ; 464 - 5 1/4 ; 465 - 5 1/4 ; 466 - 5 1/4 ; 467 - 5 1/4 ; 468 - 5 1/4 ; 469 - 5 1/4 ; 470 - 5 1/4 ; 471 - 5 1/4 ; 472 - 5 1/4 ; 473 - 5 1/4 ; 474 - 5 1/4 ; 475 - 5 1/4 ; 476 - 5 1/4 ; 477 - 5 1/4 ; 478 - 5 1/4 ; 479 - 5 1/4 ; 480 - 5 1/4 ; 481 - 5 1/4 ; 482 - 5 1/4 ; 483 - 5 1/4 ; 484 - 5 1/4 ; 485 - 5 1/4 ; 486 - 5 1/4 ; 487 - 5 1/4 ; 488 - 5 1/4 ; 489 - 5 1/4 ; 490 - 5 1/4 ; 491 - 5 1/4 ; 492 - 5 1/4 ; 493 - 5 1/4 ; 494 - 5 1/4 ; 495 - 5 1/4 ; 496 - 5 1/4 ; 497 - 5 1/4 ; 498 - 5 1/4 ; 499 - 5 1/4 ; 500 - 5 1/4 ; 501 - 5 1/4 ; 502 - 5 1/4 ; 503 - 5 1/4 ; 504 - 5 1/4 ; 505 - 5 1/4 ; 506 - 5 1/4 ; 507 - 5 1/4 ; 508 - 5 1/4 ; 509 - 5 1/4 ; 510 - 5 1/4 ; 511 - 5 1/4 ; 512 - 5 1/4 ; 513 - 5 1/4 ; 514 - 5 1/4 ; 515 - 5 1/4 ; 516 - 5 1/4 ; 517 - 5 1/4 ; 518 - 5 1/4 ; 519 - 5 1/4 ; 520 - 5 1/4 ; 521 - 5 1/4 ; 522 - 5 1/4 ; 523 - 5 1/4 ; 524 - 5 1/4 ; 525 - 5 1/4 ; 526 - 5 1/4 ; 527 - 5 1/4 ; 528 - 5 1/4 ; 529 - 5 1/4 ; 530 - 5 1/4 ; 531 - 5 1/4 ; 532 - 5 1/4 ; 533 - 5 1/4 ; 534 - 5 1/4 ; 535 - 5 1/4 ; 536 - 5 1/4 ; 537 - 5 1/4 ; 538 - 5 1/4 ; 539 - 5 1/4 ; 540 - 5 1/4 ; 541 - 5 1/4 ; 542 - 5 1/4 ; 543 - 5 1/4 ; 544 - 5 1/4 ; 545 - 5 1/4 ; 546 - 5 1/4 ; 547 - 5 1/4 ; 548 - 5 1/4 ; 549 - 5 1/4 ; 550 - 5 1/4 ; 551 - 5 1/4 ; 552 - 5 1/4 ; 553 - 5 1/4 ; 554 - 5 1/4 ; 555 - 5 1/4 ; 556 - 5 1/4 ; 557 - 5 1/4 ; 558 - 5 1/4 ; 559 - 5 1/4 ; 560 - 5 1/4 ; 561 - 5 1/4 ; 562 - 5 1/4 ; 563 - 5 1/4 ; 564 - 5 1/4 ; 565 - 5 1/4 ; 566 - 5 1/4 ; 567 - 5 1/4 ; 568 - 5 1/4 ; 569 - 5 1/4 ; 570 - 5 1/4 ; 571 - 5 1/4 ; 572 - 5 1/4 ; 573 - 5 1/4 ; 574 - 5 1/4 ; 575 - 5 1/4 ; 576 - 5 1/4 ; 577 - 5 1/4 ; 578 - 5 1/4 ; 579 - 5 1/4 ; 580 - 5 1/4 ; 581 - 5 1/4 ; 582 - 5 1/4 ; 583 - 5 1/4 ; 584 - 5 1/4 ; 585 - 5 1/4 ; 586 - 5 1/4 ; 587 - 5 1/4 ; 588 - 5 1/4 ; 589 - 5 1/4 ; 590 - 5 1/4 ; 591 - 5 1/4 ; 592 - 5 1/4 ; 593 - 5 1/4 ; 594 - 5 1/4 ; 595 - 5 1/4 ; 596 - 5 1/4 ; 597 - 5 1/4 ; 598 - 5 1/4 ; 599 - 5 1/4 ; 600 - 5 1/4 ; 601 - 5 1/4 ; 602 - 5 1/4 ; 603 - 5 1/4 ; 604 - 5 1/4 ; 605 - 5 1/4 ; 606 - 5 1/4 ; 607 - 5 1/4 ; 608 - 5 1/4 ; 609 - 5 1/4 ; 610 - 5 1/4 ; 611 - 5 1/4 ; 612 - 5 1/4 ; 613 - 5 1/4 ; 614 - 5 1/4 ; 615 - 5 1/4 ; 616 - 5 1/4 ; 617 - 5 1/4 ; 618 - 5 1/4 ; 619 - 5 1/4 ; 620 - 5 1/4 ; 621 - 5 1/4 ; 622 - 5 1/4 ; 623 - 5 1/4 ; 624 - 5 1/4 ; 625 - 5 1/4 ; 626 - 5 1/4 ; 627 - 5 1/4 ; 628 - 5 1/4 ; 629 - 5 1/4 ; 630 - 5 1/4 ; 631 - 5 1/4 ; 632 - 5 1/4 ; 633 - 5 1/4 ; 634 - 5 1/4 ; 635 - 5 1/4 ; 636 - 5 1/4 ; 637 - 5 1/4 ; 638 - 5 1/4 ; 639 - 5 1/4 ; 640 - 5 1/4 ; 641 - 5 1/4 ; 642 - 5 1/4 ; 643 - 5 1/4 ; 644 - 5 1/4 ; 645 - 5 1/4 ; 646 - 5 1/4 ; 647 - 5 1/4 ; 648 - 5 1/4 ; 649 - 5 1/4 ; 650 - 5 1/4 ; 651 - 5 1/4 ; 652 - 5 1/4 ; 653 - 5 1/4 ; 654 - 5 1/4 ; 655 - 5 1/4 ; 656 - 5 1/4 ; 657 - 5 1/4 ; 658 - 5 1/4 ; 659 - 5 1/4 ; 660 - 5 1/4 ; 661 - 5 1/4 ; 662 - 5 1/4 ; 663 - 5 1/4 ; 664 - 5 1/4 ; 665 - 5 1/4 ; 666 - 5 1/4 ; 667 - 5 1/4 ; 668 - 5 1/4 ; 669 - 5 1/4 ; 670 - 5 1/4 ; 671 - 5 1/4 ; 672 - 5 1/4 ; 673 - 5 1/4 ; 674 - 5 1/4 ; 675 - 5 1/4 ; 676 - 5 1/4 ; 677 - 5 1/4 ; 678 - 5 1/4 ; 679 - 5 1/4 ; 680 - 5 1/4 ; 681 - 5 1/4 ; 682 - 5 1/4 ; 683 - 5 1/4 ; 684 - 5 1/4 ; 685 - 5 1/4 ; 686 - 5 1/4 ; 687 - 5 1/4 ; 688 - 5 1/4 ; 689 - 5 1/4 ; 690 - 5 1/4 ; 691 - 5 1/4 ; 692 - 5 1/4 ; 693 - 5 1/4 ; 694 - 5 1/4 ; 695 - 5 1/4 ; 696 - 5 1/4 ; 697 - 5 1/4 ; 698 - 5 1/4 ; 699 - 5 1/4 ; 700 - 5 1/4 ; 701 - 5 1/4 ; 702 - 5 1/4 ; 703 - 5 1/4 ; 704 - 5 1/4 ; 705 - 5 1/4 ; 706 - 5 1/4 ; 707 - 5 1/4 ; 708 - 5 1/4 ; 709 - 5 1/4 ; 710 - 5 1/4 ; 711 - 5 1/4 ; 712 - 5 1/4 ; 713 - 5 1/4 ; 714 - 5 1/4 ; 715 - 5 1/4 ; 716 - 5 1/4 ; 717 - 5 1/4 ; 718 - 5 1/4 ; 719 - 5 1/4 ; 720 - 5 1/4 ; 721 - 5 1/4 ; 722 - 5 1/4 ; 723 - 5 1/4 ; 724 - 5 1/4 ; 725 - 5 1/4 ; 726 - 5 1/4 ; 727 - 5 1/4 ; 728 - 5 1/4 ; 729 - 5 1/4 ; 730 - 5 1/4 ; 731 - 5 1/4 ; 732 - 5 1/4 ; 733 - 5 1/4 ; 734 - 5 1/4 ; 735 - 5 1/4 ; 736 - 5 1/4 ; 737 - 5 1/4 ; 738 - 5 1/4 ; 739 - 5 1/4 ; 740 - 5 1/4 ; 741 - 5 1/4 ; 742 - 5 1/4 ; 743 - 5 1/4 ; 744 - 5 1/4 ; 745 - 5 1/4 ; 746 - 5 1/4 ; 747 - 5 1/4 ; 748 - 5 1/4 ; 749 - 5 1/4 ; 750 - 5 1/4 ; 751 - 5 1/4 ; 752 - 5 1/4 ; 753 - 5 1/4 ; 754 - 5 1/4 ; 755 - 5 1/4 ; 756 - 5 1/4 ; 757 - 5 1/4 ; 758 - 5 1/4 ; 759 - 5 1/4 ; 760 - 5 1/4 ; 761 - 5 1/4 ; 762 - 5 1/4 ; 763 - 5 1/4 ; 764 - 5 1/4 ; 765 - 5 1/4 ; 766 - 5 1/4 ; 767 - 5 1/4 ; 768 - 5 1/4 ; 769 - 5 1/4 ; 770 - 5 1/4 ; 771 - 5 1/4 ; 772 - 5 1/4 ; 773 - 5 1/4 ; 774 - 5 1/4 ; 775 - 5 1/4 ; 776 - 5 1/4 ; 777 - 5 1/4 ; 778 - 5 1/4 ; 779 - 5 1/4 ; 780 - 5 1/4 ; 781 - 5 1/4 ; 782 - 5 1/4 ; 783 - 5 1/4 ; 784 - 5 1/4 ; 785 - 5 1/4 ; 786 - 5 1/4 ; 787 - 5 1/4 ; 788 - 5 1/4 ; 789 - 5 1/4 ; 790 - 5 1/4 ; 791 - 5 1/4 ; 792 - 5 1/4 ; 793 - 5 1/4 ; 794 - 5 1/4 ; 795 - 5 1/4 ; 796 - 5 1/4 ; 797 - 5 1/4 ; 798 - 5 1/4 ; 799 - 5 1/4 ; 800 - 5 1/4 ; 801 - 5 1/4 ; 802 - 5 1/4 ; 803 - 5 1/4 ; 804 - 5 1/4 ; 805 - 5 1/4 ; 806 - 5 1/4 ; 807 - 5 1/4 ; 808 - 5 1/4 ; 809 - 5 1/4 ; 810 - 5 1/4 ; 811 - 5 1/4 ; 812 - 5 1/4 ; 813 - 5 1/4 ; 814 - 5 1/4 ; 815 - 5 1/4 ; 816 - 5 1/4 ; 817 - 5 1/4 ; 818 - 5 1/4 ; 819 - 5 1/4 ; 820 - 5 1/4 ; 821 - 5 1/4 ; 822 - 5 1/4 ; 823 - 5 1/4 ; 824 - 5 1/4 ; 825 - 5 1/4 ; 826 - 5 1/4 ; 827 - 5 1/4 ; 828 - 5 1/4 ; 829 - 5 1/4 ; 830 - 5 1/4 ; 831 - 5 1/4 ; 832 - 5 1/4 ; 833 - 5 1/4 ; 834 - 5 1/4 ; 835 - 5 1/4 ; 836 - 5 1/4 ; 837 - 5 1/4 ; 838 - 5 1/4 ; 839 - 5 1/4 ; 840 - 5 1/4 ; 841 - 5 1/4 ; 842 - 5 1/4 ; 843 - 5 1/4 ; 844 - 5 1/4 ; 845 - 5 1/4 ; 846 - 5 1/4 ; 847 - 5 1/4 ; 848 - 5 1/4 ; 849 - 5 1/4 ; 850 - 5 1/4 ; 851 - 5 1/4 ; 852 - 5 1/4 ; 853 - 5 1/4 ; 854 - 5 1/4 ; 855 - 5 1/4 ; 856 - 5 1/4 ; 857 - 5 1/4 ; 858 - 5 1/4 ; 859 - 5 1/4 ; 860 - 5 1/4 ; 861 - 5 1/4 ; 862 - 5 1/4 ; 863 - 5 1/4 ; 864 - 5 1/4 ; 865 - 5 1/4 ; 866 - 5 1/4 ; 867 - 5 1/4 ; 868 - 5 1/4 ; 869 - 5 1/4 ; 870 - 5 1/4 ; 871 - 5 1/4 ; 872 - 5 1/4 ; 873 - 5 1/4 ; 874 - 5 1/4 ; 875 - 5 1/4 ; 876 - 5 1/4 ; 877 - 5 1/4 ; 878 - 5 1/4 ; 879 - 5 1/4 ; 880 - 5 1/4 ; 881 - 5 1/4 ; 882 - 5 1/4 ; 883 - 5 1/4 ; 884 - 5 1/4 ; 885 - 5 1/4 ; 886 - 5 1/4 ; 887 - 5 1/4 ; 888 - 5 1/4 ; 889 - 5 1/4 ; 890 - 5 1/4 ; 891 - 5 1/4 ; 892 - 5 1/4 ; 893 - 5 1/4 ; 894 - 5 1/4 ; 895 - 5 1/4 ; 896 - 5 1/4 ; 897 - 5 1/4 ; 898 - 5 1/4 ; 899 - 5 1/4 ; 900 - 5 1/4 ; 901 - 5 1/4 ; 902 - 5 1/4 ; 903 - 5 1/4 ; 904 - 5 1/4 ; 905 - 5 1/4 ; 906 - 5 1/4 ; 907 - 5 1/4 ; 908 - 5 1/4 ; 909 -

22	3	TIME	Heure de création ou de dernière modification du fichier
			OCIEI 23 OCIEI 22
			H H H H H H H H H H S S S S
26	1	DEVIS	Devis de l'identification du périphérique au lequel la lecture se fera, format : O O O O O O O O O O = No du Groupe IO-1 I C I I P P P P C C P P P P = Caractéristique P = COM (COMmode) G = GUC (GUCmode) O = NUL (NUL device) C = LBY (LIST = Input/output) A = PRN (Printer) C = C C (C C mode) depuis l'Espan. E = I O (I O mode) sans pas C I modification
25	1	DIRECT	Directory location : position du fichier dans le Directory sous le nom d'un cluster d'entité (OO - AF I).
26	2	SYNCL	Start Cluster : donne le bloc physique à partir duquel la lecture du fichier.
28	2	CURCL	Current Cluster : en bloc physique le cluster actuel sous le nom d'un cluster.
30	2	CLOFF	Cluster Offset : en bloc physique le dernier cluster excédant l'adresse du début du fichier.
32	1	CP	Current Record : no du record 10-1271 ou l'union de l'adresse. Cette position est calculée à partir de la position de chaque cluster physique. P 128, 010 est l'adresse de 0 et 22 est l'adresse de 1.
33	3	HN	Record Number : décodeur pour les fichiers directs. Contient le numéro du record à accéder, l'octet 35 est toujours 0.

Dans l'écriture programmeur, on accède à un fichier, il faut d'abord indiquer la première octa du FCB avec le numéro du fichier, le nom du fichier, son extension et généralement compléter le tout de la octa des 004.

Ensuite, il procède à l'ouverture du fichier.

Pour accéder aux informations d'un fichier quelconque, il lui suffit d'utiliser les fonctions CP/R READ SEQUENTIAL I/O et WRITE SEQUENTIAL I/O sans intervenir sur FCB, à l'exception de la mise enregistrement de la donnée.

Si l'octa accède aux informations hors séquence, il doit indiquer le numéro de l'opération de lecture ou d'écriture et le numéro de l'octet de l'octet à l'avant chaque READ ou WRITE SEQUENTIAL.

Pour accéder aux informations d'un fichier à accès direct, il doit passer le numéro de l'enregistrement dans

les octets 10 à 31 peuvent être utilisés par le programmeur pour des modifications, ils ne sont pas utilisés par les CP/R pour les opérations de lecture ou d'écriture. Cependant, le programmeur procède à la lecture du fichier.

Les octets 10 à 31 peuvent être utilisés par le programmeur pour des modifications, ils ne sont pas utilisés par les CP/R pour les opérations de lecture ou d'écriture. Cependant, le programmeur procède à la lecture du fichier.

3.2.2 Le FCB du MSX-DOS et du Disk-Basic

Le FCB du MSX-DOS est complètement identique au FCB du MS-DOS qui est le système d'exploitation des IBM-PC et des compatibles et la différence est qu'il ne supporte pas le "Random Access" ou le "Direct Access".

Il est constitué de 37 octets, dont 10 à 31 sont dans le FCB CP/R, les autres octets sont utilisés d'ailleurs dans les deux types de FCB. Dans le cas d'un fichier, les octets d'adresse de chaque cluster, les octets de la longueur du FCB sont indiqués.

Zone RECORD : Position : 13 Longueur : 2 octets

Pour les fichiers MSX-DOS 2.0 et 2.1, on choisit la longueur du fichier en indiquant dans le FCB la longueur du fichier. Cette valeur est la même que la longueur du fichier.

Zone NC : Position : 33 Longueur : 1 octet

Record Number : Cette zone est utilisée par les fonctions MSX-DOS 2.0 et 2.1 et par les fonctions 2.0 et 2.1 du CP/R. Dans le cas d'un fichier, elle est toujours 0. Les octets 10 à 31 sont les octets de la longueur du fichier. Les octets 10 à 31 sont les octets de la longueur du fichier. Les octets 10 à 31 sont les octets de la longueur du fichier. Les octets 10 à 31 sont les octets de la longueur du fichier.

Dans le FCB MSX-DOS le cluster est toujours 0. Le FCB est identique au FCB CP/R pour les fonctions MSX-DOS d'accès séquentiel.

Pour les fichiers Disk-Basic non compatibles CP/R, le FCB est identique au FCB CP/R pour les fonctions MSX-DOS d'accès séquentiel. Il est identique au FCB CP/R pour les fonctions MSX-DOS d'accès séquentiel.

En effet, si on peut lire une longueur d'enregistrement de 1 octet, alors les octets de la longueur d'enregistrement sont les octets de la longueur d'enregistrement.

Les points d'entrée du Disk-ROM

Il existe actuellement toute une série de contrôleurs de disquettes. On peut cependant les classer en 4 grandes catégories:

- a) Les contrôleurs de disquettes 360K datent du MSX1
- b) Les contrôleurs de disquettes 720K datent du MSX1
- c) Les contrôleurs de disquettes 360K datent du MSX2
- d) Les contrôleurs de disquettes 720K datent du MSX2

La différence essentielle entre les versions MSX1 et MSX2 se trouve dans le fait que de nouveaux points d'entrée ont été ajoutés et que des erreurs mineures ont été corrigées dans la version MSX2.

A part cela, les versions 360K appellent également quelques remarques. On les appelle versions 360K parce qu'elles sont fournies avec une unité de disquette de 360K, or certaines d'entre elles supportent également la connexion de disquettes 720K alors que d'autres ne les supportent pas. Il est donc primordial de pouvoir distinguer ces deux versions et d'en connaître les avantages et inconvénients.

Les versions 360K qui affichent 24456 ou 24455 octets libres lors d'un PRINT FRE(0) sont des versions qui ne supportent pas les disquettes de 720K. Par contre, les versions qui affichent 23432 octets libres après un PRINTFRE(0) supportent les disquettes de 720K.

Les avantages et inconvénients des deux versions sont exactement opposés : l'un ne supporte pas les disquettes de 720K mais offre 1024 octets de plus et vice-versa pour l'autre.

En dehors de cela, il est un aspect plus difficile à identifier: c'est la vitesse effective de lecture et d'écriture sur la disquette. Elle ne dépend pas du classement dans l'une des quatre catégories précédentes mais du fabricant de ce contrôleur.

En effet, rappelons ici que la ROM interne d'un contrôleur contient trois modules:

- Le Disk-Basic1
- Le Kernel MSX-DOS, programmé par MICROSOFT;
- Le Driver (pilote) du lecteur, programmé par le fabricant de ce contrôleur.

Or la vitesse dépend essentiellement de ce Driver et donc varie d'un fabricant à l'autre; reconnaissons cependant que les versions MSX2 sont, généralement, plus rapides que les versions MSX1.

Pour pouvoir utiliser les points d'entrée du Disk-ROM qui

est situé en Page 1 c'est-à-dire à l'adresse 4000H, il faut tout d'abord sélectionner le slot où est situé le ROM. D'autre part, si nous avons plusieurs contrôleurs, ils seront nécessairement disposés dans des slots différents et demanderont donc une sélection spécifique pour pouvoir manipuler les unités de disquettes de ces contrôleurs.

Comment donc sélectionner le Disk-ROM qui nous intéresse? Heureusement pour nous, une zone de mémoire RAM a été initialisée à l'usage du système pour nous indiquer dans quel slot se trouve chacun des quatre contrôleurs possibles et combien de disquettes logiques ils manipulent chacun.

Cette zone de mémoire est située de l'adresse FB21H à FB26H. Bien entendu cette zone n'est valide que si votre système contient au moins un contrôleur. Pour rendre vos programmes compatibles sur tout système, il suffit de vérifier que la Hook FFA7H ne contient pas la valeur C9H; dans ce cas, aucun contrôleur n'est installé.

Table des Contrôleurs

FB21 : Nombre d'unités logiques connectées au contrôleur 0
 FB22 : Slot où est situé le contrôleur 0 sous la forme :
 E 0 0 0 B S P P où PP = slot primaire
 SS = slot secondaire si E = 1

FB23 : Nombre d'unités logiques connectées au contrôleur 1
 FB24 : Slot où est situé le contrôleur 1

FB25 : Nombre d'unités logiques connectées au contrôleur 2
 FB26 : Slot où est situé le contrôleur 2

FB27 : Nombre d'unités logiques connectées au contrôleur 3
 FB28 : Slot où est situé le contrôleur 3

Suivant l'unité que nous voulons atteindre (A:=0 B:=1 C:=2... N:=7), il suffit maintenant de trouver quel contrôleur nous devons sélectionner en scrutant le numéro de l'unité choisie le nombre de lectures du CTRL0, puis du CTRL1, etc jusqu'à ce que le résultat soit plus petit que 1, et ensuite de sélectionner le contrôleur en tenant compte du slot où il est situé. Nous ferons pour cela appel à la routine de sélection de slots de la ROM-BIOS à l'adresse 24H.

Il existe d'autres méthodes pour appeler une routine donnée dans un slot choisi (voyez pour le chapitre 9), mais dans la mesure du possible nous emploierons la méthode de numérotation de slots (0024H) plutôt que d'autres méthodes, pour des raisons didactiques.

Voici une petite routine qui vous permettra de sélectionner le slot correspondant à l'unité logique dont vous aurez mis le numéro dans l'accumulateur 10-71.

En sortie, la Page 1 (4000H) sélectionnera le slot du contrôleur approprié et l'accumulateur donnera le numéro de l'unité physique de ce contrôleur 10-11.

```
RPT: LD HL,FB21H
      SUB HL,1
      JR C,FBI
      INC HL
      INC HL
      JR RPT
FBI: ADD A,HL
      PUSH AF
      INC HL
      LD A,HL
      LD H,40H
      CALL 0024H
      EI
      POP AF
```

Bien entendu, si vous n'avez qu'un seul contrôleur, la tâche sera simplifiée puisque le numéro du contrôleur est connu 101, il suffit donc de sélectionner ce contrôleur par :

```
LD A,10FB22H
LD H,40H
CALL 0024H
EI
```

Le position FB46 donne également le numéro de slot du contrôleur 0.

Ce point d'entrée permet de lire ou d'écrire de 1 à 255 secteurs en un appel de la routine à partir du secteur spécifié sur une unité de disquettes de type et de numéro déterminés.

En entrée

CF La carry flag est mise à 1 pour lire
La carry flag doit être à 1 pour écrire

A Numéro de l'unité logique choisie (0 à 1) du contrôleur dont le ROM est actuellement sélectionné. Un contrôleur ne comporte que deux unités qui sont toujours numérotées 0 et 1.

B Nombre de secteur (1 à 255) à transférer d'une fois le secteur. Ce nombre doit tenir compte de la taille de la mémoire réservée pour allouer les secteurs. En effet, si vous demandez le transfert de 100 secteurs de 512 octets, il faudra 51200 positions de secteur pour les allouer et vous devrez donc avoir prévu la place nécessaire.

C Code du type de disquette IFB à FF voir annexe A1. Attention! certains contrôleurs récents ne supportent plus les types FC à FF réservés pour les disquettes de 5 1/4.

DE Numéro du premier secteur à transférer (voir annexe A pour les numéros de secteur suivants les types de disques pour un disque de 720K; la gamme s'étend de 0 à 255. Neuf pour un disque de 720K, la gamme s'étend de 0 à 599 Neuf).

HL Contient l'adresse de la zone mémoire de réception/emission des secteurs. Les secteurs lue sont envoyés à l'adresse HL ou le contenu de l'adresse HL sera écrit sur le disque suivant l'état du Carry Flag.

En sortie

CF 0 si pas d'erreur.
1 si l'écriture/lecture est erronée

A Si le Carry Flag est 1, A contient un des codes d'erreur suivants:

0 1 La disquette est protégée contre l'écriture lors d'une tentative d'écriture (Write protect)

2 1 Pas de disquette dans le lecteur ou lecteur non allumé ou lecteur inexistante (Disk Missing)

4 1 Mauvaise lecture des données (CRC error)

6 1 Erreur de positionnement de la tête de lecture (Seek error)

B 1 Secteur non trouvé (Header not found) nécessite un reformatage.

A 1 Mauvaise écriture (Write error) or (RAM error)

C 1 Autres types d'erreur non spécifiés (ex: DMA line out)

Regardez électrons AF, BC, DE, HL, IX, IY

Attention! Il s'agit ici de lecture ou d'écriture physique, l'adressage du disque se fait par numéro de secteur et non par numéro d'adressage dans un fichier. D'autre part, ce point d'entrée est éventuellement remplacé par la CML DISK 1441 de la ROM Basic où tous les paramètres sont identiques sauf A qui représente les unités logiques de 0 à 7. En effet, la routine détermine elle-même quel contrôleur choisir d'après le numéro de disque 0 à 7.

Exemple:

Lire les secteurs 5 à 11 (Directory) du disque C: 1350K1 en CIOH de la mémoire. Nous intégrerons cette routine dans un programme Basic pour alléger tous les fichiers en Directory.

		ORG	COOQ	
		C006 3E D2	LD A, 2	A = 2 = Disque C:
		C007 21 D1 FB	LD HL, 0FB21	Table contrôleurs
		C008 96	SHB HL	
		C009 38 06	JR C, CONT	
		C00A 23	INC HL	
		C00B 23	INC HL	Sélectionne
		C00C A8 F9	JR PPT	le slot du contr.
		C00D 84	ADD A, HL	du disque C:
		C00E F3	PUSH AF	A indique si C est
		C00F 23	INC HL	le premier 101 du
		C010 26 40	LD A, (HL)	le second disque
		C011 C9 74 00	LD H, 40	
		C012 FB	CALL 0074	
		C013 F1	POP AF	
		C014 06 07	LD B, 7	Lecture de 7 sect.
		C015 0E FB	LD C, 0FB	à partir du sect 5
		C016 11 05 00	LD DE, 5	en CIOH.
		C017 21 00 C1	LD HL, 0C100	
		C018 A7	AND A	
		C019 D0 10 40	CALL 0401D	
		C020 21 FF 00	LD HL, 0FF	pas d'erreur = 255
		C021 30 01	JR NC, REND	variable basic.
		C022 4F	LD A, 0	code d'erreur
		C023 22 F8 F7	LD HL, 0FF8F8	-> variable basic
		C024 3E 02	LD A, 2	
		C025 52 43 F6	LD HL, 0F643A	
		C026 3A C1 FC	LD A, 0FCFC1	Sélectionne Row
		C027 26 40	LD H, 40	base et retour au
		C028 C9 24 00	CALL 0024	base
		C029 FB	EI	
		C030 C9	RET	

Pour exploiter cette petite routine dans le programme Base1, vous compilerez comme suit les lignes 20 à 35. A un seul disque, modifier l'instruction de l'adresse 0000 par 3E 00 pour lire le Directory du lecteur A; plutôt que celui du lecteur C; Dans le programme Base1, remplacer le 3E 02 de la ligne 01A quatre 50 par 3E 001.

```

10 CLEAR200, $H0FFF
20 CLS:AD=84000:DEFUSR=AD
30 FORI=ND01:60:READAS
40 PERFI,VOL:IAH:IAEI:INEI
50 DATA 3E,02,21,21,FB,66,38,01,23,23,10,F9
60 DATA 84,FB,23,7F,26,10,C0,21,00,FB,F1,06
70 DATA 07,06,FB,11,05,00,21,00,C1,A7,C0,10
80 DATA 40,21,FB,00,30,01,6F,22,FB,F7,C0,02
90 DATA 37,A3,FB,3A,C1,FB,26,10,C0,24,00,18,C9
100 AS=USRIOI
110 IF ATC256010160
120 FORJ=HIC0010171512:JSTP32
130 FORJ=010101C=PERFI+31:IFC=01HENENILOCIFC=HIC0010150
140 PRINTCHR(1C1)+NEI:J,FAINT
150 NEXTJ:END
160 ON R/3 GOTO 170,180,1A0,200,210,220:END
170 PRINT'DISK OFFLINE':END
180 PRINT'DISK READ ERROR':END
190 FAINT'DISK SEEK ERROR':END
200 PRINT'BLOCK NOT FOUND':END
210 ASH
220 PRINT'DMA TIME OUT':END

```

Une autre méthode consiste à employer un point d'entrée de la ROM-Bios du Basic plutôt que le point d'entrée 4010H du contrôleur. Le programme de l'organe achève sera nettement plus court parce que la routine Au Bios va elle-même sélectionner la commande adéquate à l'adresse la plus élevée du programme en ayant les restrictions de la ROM-Bios. L'emploi des registres est identique excepté que le registre A peut varier de 0 A 7 au lieu de 0 à 1. (Hill A1 A11).

	ORG	CODS	
0000	3E 07	LD A,2	disque logique C1
0002	06 07	LD B,7	abr secteur A lire
0004	06 08	C 0,FB	type de disque
0006	11 05 00	LD DE,0005	secteur départ = 5
0008	21 00 C1	LD HL,0C100	adresse l'opération
000C	A7	AND A	carry = 0 = lire
000D	00 14 01	CALL 0144	appel ROM Bios 144
0010	21 FF 00	LD HL,00FF	si pas erreur, 255
0013	30 01	JR NC,ND005	variable A=0
0015	AF	LD A	code d'erreur A=0
0016	22 FB F7	RDEND: LD 10FFFB,1A	variable basc
0019	3E 02	LD A,7	1
001B	32 03 F6	LD 10F631,A	1
001E	C9	RET	retour au Basic

Si vous intégrez cette routine dans le programme Base1 et décrivez, modifiez les lignes 20 à 35 le contenu du langage assemblé de la nouvelle routine et remplacez 3E de charger le ligne 30 par :

```
30 FORI=ND01:30:DEFUSR=AD
```

5.2 0013H DISQUE D:SK CHENGE OVER

Cette routine vérifie si le disque est bien chargé. Il est important, lorsqu'un programme lui est écrit sur une disquette, de vérifier si cette disquette est bien remplie par son contenu et l'adresse du programme, la réponse de la routine peut être l'un des 5 codes suivants :

- 01 Le disque est bien chargé.
- 02 Le disque est bien chargé.
- 03 Le disque est bien chargé.
- 04 Le disque est bien chargé.
- 05 Le disque est bien chargé.

Dans les cas 01 et 02, la routine va éliminer le type de disquette IFB A FF1 placée dans le lecteur et va libérer le DPM Disk Controller Block voir chapitre 5.4) correspondance au type de l'ancienne nouvelle disquette. Ce type sans correspondance en lisant le premier secteur du secteur 1 (FAT) ou la position 27 du secteur 0 (Boot Sector), suivant les caractéristiques. Ce caractère est appelé le 'Media Descriptor' et A déjà été défini dans les chapitres 5.1 (DPM) et 1.5.

En entrée :

- A Numéro du disque logique (0 ou 1) de ce contrôleur
- B 0
- C Type de disque que l'on s'efforce de trouver dans le lecteur IFB et FRI. H = Adresse du DPM du disque logique voir chapitre 5.4

En sortie :

- CF = 0 L'opération a été terminée sans erreur
- CF = 1 Le disque est bien chargé
- 0 = FF Le disque est bien chargé et le type de disque est identifié
- B = 0 Le disque est bien chargé

Dans le cas où le disque est bien chargé et qu'il est bien chargé, le DPM peut lire le 1A et 1B et A peut avoir les paramètres correspondants au type de la disquette présente dans le lecteur.

- CF = 1 Une erreur s'est produite pendant l'opération
- A = 2 Fin de disquette dans le lecteur
- A = 3 Mauvaise lecture du secteur 0 ou 1
- A = 4 Mauvaise positionnement sur le piste 0
- A = 5 Fin trouvé le secteur 0 ou 1
- A = 0A Le type de la disquette n'est pas supporté par le lecteur
- A = 0C Autres types d'erreur (DMA) lire ou 1

Registres utilisés : AF, BC, DE, HL, 11, 17

Exemples :

Vous avez un lecteur de disquette double face. Vous lisez une disquette de type INCONNU. Avant de lire ou d'écrire sur cette disquette, il faut charger le DPM avec les paramètres appropriés au type de disquette. D'autre part, il est

important de connaître le type de disquette afin de savoir où se trouve le Directory, par exemple.

Voici un petit programme qui vous renseignera le type de disquette insérée dans le lecteur A: et si cette disquette a été changée oui ou non.

```

ORC      C000
C000 3A 22 F8 START: LD  A,10FB221 ; Sélectionnons le ROM
C003 26 40 LD  M,40 ; du contrôleur 0
C005 CD 24 00 CALL 0024 ;
C00B 3E 00 LD  A,0 ; Disque A:
C00A 01 F8 00 LD  BC,00F8 ; type attendu 360K
C00D 2A 35 F3 LD  DE,10F3551 ; Pointeur DPAD
C010 CD 13 40 CALL 4013 ; Cell DEKCHG
C013 26 00 LD  M,0 ;
C015 6F LD  L,A ; Code erreur -> HL
C016 3E 03 JR  C,REND ; Si erreur -> REND
C018 05 DEC  B ;
C019 05 DEC  B ; Si ok, B-2 -> HL
C01A 68 LD  L,B ;
C01D 22 F8 F7 REND: LD  10F7F01,HL ; HL -> variable
C01E 3E 02 LD  A,2 ; DABIC
C020 32 63 F6 LD  10F6631,A ; de type 360K
C023 3A C1 FC LD  A,10FCC11 ; Rétablit Rom Basic
C026 26 40 LD  M,40 ;
C02B CD 24 00 CALL 0024 ; et
C028 F8 EI ; Exit vers Basic
C02C C9 RET ;

```

Incorporez cette routine dans le programme Basic:

```

10 CLEAR200,10000
20 CLS:AD=10000:DEFUSR=AD
30 FOR I=ADTO1:44:READA$
40 FOR K=1,VAL ("A"-"A")+1:NEXT
50 DATA 3A,22,F8,26,40,CD,24,00,3E,00,01,F8,00
60 DATA 2A,35,F3,CD,13,40,26,00,6F,3E,03,05,05
70 DATA AB,22,F8,F7,3E,02,32,63,F6,3A,C1,FC,26
80 DATA 10,CD,24,00,F8,C9
90 AX=USR10
95 DPB=PEEK(10F3551)+256:PEEK(10F356):TS=JES:PEEK(DPB+11)
100 IF AX<256GOTO150
110 DNAX=256GOSUB120,130,140:GOTO90
120 PRINT"Disque changé. Nouveau type = "TS:RETURN
130 PRINT"Disque peut-être changé. Type = "TS:RETURN
140 PRINT"Disque non changé. Type = "TS:RETURN
150 DN AX/2 GOSUB160,170,180,190,200,210:GOTO90
160 PRINT"DISK OFFLINE":RETURN
170 PRINT"SEEK ERROR":RETURN
180 PRINT"SECTOR 0/1 READ ERROR":RETURN
190 PRINT"SECTOR 0/1 NOT FOUND":RETURN
200 PRINT"DISK TYPE NOT SUPPORTED":RETURN
210 PRINT"DMA TIME OUT":RETURN

```

Il est à noter que le message "Disque changé..." n'apparaît que si le type de la disquette est différent de celui de la disquette précédemment installée dans le lecteur. Il faut aussi noter que les routines du "Disk Driver" varient parfois d'un fabricant à l'autre ou d'une version de "Disk Driver" à l'autre chez un même fabricant.

Le principe généralement utilisé par la majorité des fabricants pour déterminer si un disque a été changé est de lancer un chronomètre à la fin de chaque opération disque. Si le chronomètre n'a pas atteint le temps de 0,7 seconde au moment où la routine DEKCHG s'exécute, la disquette ne peut pas avoir été changée (testée donc l'écart de changer la disquette en moins de 0,7 seconde !!!).

Si, par contre, le chronomètre a dépassé le temps de 0,7 seconde, alors le secteur 0 du lecteur par la routine et si le type de disque ainsi détecté est différent de celui que vous avez renseigné dans le registre C, c'est que la disquette a été changée, même si la routine déclare simplement que la disquette pourrait peut-être avoir été changée.

6.3 4016H GETDPB GET Disk Parameter Block

Cette routine permet d'installer les paramètres d'un type de disquette dans le DPB (Disk Parameter Block), spécifié par HL.

Vous avez, par exemple, un MSX2 V68235 de PHILIPS avec un disque de 320K intégré (Type FB). Si vous y connectez un disque 5 1/4 de 320K (Type FA) au connecteur pour second disque à l'arrière de votre console, il faudra, avant de pouvoir utiliser ce second lecteur, installer les paramètres de ce type de disquette. C'est précisément ce dont s'occupe cette routine.

En entrée:

- A Numéro de l'unité physique choisie (0 ou 1) de ce contrôleur
- B Code du type de disquette (FB à FF) dont on souhaite installer les paramètres dans le DPB
- HL Adresse du DPB de l'unité choisie

En sortie:

Le DPB est mis à jour avec les paramètres du type de disquette spécifié par B. Le DPB commence à l'adresse précisée par HL, mais seules les positions HL+1 à HL+18 sont mises à jour.

Registres affectés: AF, BC, DE, HL, IX, IY

Cette routine est implicitement appelée par la routine DSKCHK-4013H décrite ci-avant lorsque le disque A ou pourrait avoir été changé. En conséquence il ne sera pas proposé d'exemples pour ce point d'entrée.

6.4 4019H CHOICE of type of format

Cette routine est appelée par les routines FORMTH (4025H) et FORMTK (4026H) qui sont respectivement la routine de formatage de la disquette en MSX-DDS et une routine de formatage générale. Toutes deux peuvent proposer des choix de format à l'écran.

Ce point d'entrée fournit simplement dans HL l'adresse du message de choix de format. En effet, les fabricants de contrôleurs disque n'offrent pas nécessairement tous les types de format. Ainsi, par exemple, le contrôleur intégré du MSX2 V68235 de PHILIPS n'autorise aucun choix lorsqu'il s'agit de formater une disquette sur le lecteur intégré mais en offre un lorsqu'il s'agit de formater une disquette sur le lecteur externe 1360K ou 720Ki. Dans le cas de cet ordinateur, le message de choix:

- 1- Single Side ...
- 2- Double Side ...

En entrée Rien

En sortie

HL 0000H s'il n'y a pas de message de choix
HL Adresse du message de choix de format

Le message de choix indique toujours un numéro. En effet, ce numéro servira dans la routine de formatage à sélectionner le format désiré. Le message de choix se termine toujours par 00H.

Registres affectés :

Tous les registres peuvent être affectés bien que la majorité des fabricants les préservent tous excepté HL, bien entendu.

Ce point d'entrée est automatiquement appelé par les deux routines décrites ci-dessus. Son utilité pour nous est donc minime sauf dans le but d'analyser le message par programme pour connaître les différents formats supportés.

C'est le point d'entrée de la routine de formatage proprement dite. Toutes les pistes vont être formatées, le secteur D (secteur) va être installé, les deux copies du FAT initialisées à 00H, sauf les trois premiers bytes qui contiendront le type de disque IFB & FFI et deux octets FFH. Les secteurs réservés du Directory vont être initialisés à 00H et tous les autres secteurs à E5H.

En entrée:

A Code de type de format (1 à 6). Si la routine s'offre pas de choix de format, A a pas d'effet. Sinon, la tête de choix doit correspondre au numéro du choix présent dans le message dont l'adresse est donnée par le point d'entrée précédent (CHOICE 00)H.

B Numéro de l'unité physique à formater (0 ou 1).

HL Adresse d'une zone de travail que la routine peut utiliser sans risquer d'effacer votre programme.

BC Longueur de cette zone de travail.

En sortie:

CF Si l'opération se déroule sans erreur, CF = 0.

Si une erreur se produit, CF = 1 et A donne le code d'erreur dont la signification est reprise ci-dessous.

A = 00 La disquette est protégée contre l'écriture.
A = 02 Il n'y a pas de disquette dans le lecteur.
A = 04 Erreur de lecture.
A = 06 Erreur de positionnement des têtes.
A = 08 Secteur pas trouvé.
A = 0A Erreur d'écriture.
A = 0C Mauvaise paramètre d'entrée (A ou B).
A = 0E Zone de travail trop petite.
A = 10 Autre type d'erreur (DMA line out).

Registres affectés: AF, BC, DE, HL, IX, IY.

Ce point d'entrée a été intéressant quo à vous conseillez d'avance le numéro du disque où sera insérée la disquette à formater et le code du type de format désiré. L'avantage de cette routine par rapport au deux autres est qu'elle laisse l'erreur intacte.

Exemples:

Dans un V08235 de PHILIPS, le second lecteur du contrôleur intégré peut être formaté suivant le choix 1 (360K) ou suivant le choix 2 (720K). Voici une routine qui va formater la disquette du disque B1 au format 720K, pour autant que le lecteur B1 soit du type "Double Face" bien entendu. Nous l'intégrons dans un programme Basic:

DWG 0000

```
0000 3A 49 F3 START: LD A,(OF34B) ; Select Rom diaque
0003 26 40 LD H,00 ;
0005 CD 20 00 CALL 0024 ;
0008 3E 02 LD A,2 ; Format 2 1720K
000A 16 01 LD D,1 ; Disque B1
000C D1 00 30 LD H,9000 ; Zone de travail
000E 21 00 90 LD DE,3000 ; Longueur zone
0010 CD 1C 40 CALL 001C ; Call format
0015 21 FF 00 LD H,00FF ; 255 > HL
001A 30 01 JR NC,FIN ; pas erreur > FIN
001A 4F LD L,A ; Code erreur > HL
001B 22 F8 F7 FIN: LD 1DF7F8,HL ; HL -> variable
001E 3E 03 LD A,3 ; type secteur
0020 32 43 F6 LD (0F6A3),A
0023 3A C1 FC LD A,(0FC1C) ; selection Rom Basic
0024 26 40 LD H,00
0028 C3 24 00 JP 0028 ; al retour au Basic.
```

```
10 CLC
20 CLS
30 PRINT "FORMATAGE DISQUE B1 EN 720K"
40 READ #1:POKE 1,VAL("B1")
50 READ #1:POKE 1,VAL("B1")
60 IF A=255 THEN END
70 ON A/2+1 GOTO 90,90,100,110,120,130,140,150,160
80 PRINT "Disque protégé"
90 PRINT "Erreur vide"
100 PRINT "Erreur de lecture"
110 PRINT "Erreur de positionnement"
120 PRINT "Secteur pas trouvé"
130 PRINT "Erreur d'écriture"
140 PRINT "Mauvaise paramètre"
150 PRINT "Zone de travail trop petite"
160 PRINT "Autre type d'erreur"
170 DATA 36,48,72,90,ED,25,00,3E,02,16,01,21,00,90,03,00,30
180 DATA CD,1E,00,21,FF,00,30,01,4F,22,F0,F2,3E,03,32,43,F6,3A
190 DATA E1,FC,26,40,E7,29,00
```


6.6 FORMAT FORMAT Hex-dos

Ce point d'entrée va simplement appeler le suivant 14026H
FORMAT en ayant préalablement activé le CARRY FLAG.

6.7 FORMAT FORMAT with keyboard choice

Ce point d'entrée sert au formatage d'une disquette tant en Bealé qu'en MSX DOS. Le bon déroulement d'une opération de formatage nécessite une certaine quantité de mémoire libre. Il faut donc que la routine FORMAT connaisse l'emplacement et la dimension de cette mémoire libre de travail. Il y a deux possibilités :

1) En MSX DOS, il suffit d'indiquer dans HL l'adresse de la zone de travail et dans BC la dimension de cette zone et de placer le CARRY FLAG à l'état 1. Le CF est automatiquement mis à 1 par le point d'entrée précédent -> FORMAT).

2) En Disk-BASIC, il n'est plus nécessaire de remplir HL et BC car ils vont être automatiquement trouvés par la routine. En effet, l'adresse de la zone de travail proviendra du pointeur I/O qui indique en Basic le début de la zone de mémoire libre qui s'étend jusqu'à la zone de la pila (STACK) moins 256 octets. Le I/O de la zone sera donnée par la soustraction de ces deux adresses. Pour que la routine agisse de la sorte, le CARRY FLAG doit être à l'état 0 lors de l'appel de la routine.

Que fait encore cette routine? Elle va afficher à l'écran un message vous demandant quelle unité vous désirez employer pour formater votre disquette.

Si vous avez 4 unités logiques sur votre système, les messages sera : "DRIVE NAME 1A,B,C,D". Les noms afforés en option dans les parenthèses sont automatiquement ajoutés suivant la configuration disque dont vous êtes équipé.

Lorsque vous aurez répondu au choix proposé, un éventuel message de choix de format vous sera proposé et le contrôleur du disque choisi supportera plusieurs formats (voir point d'entrée 4 -> 4010 -> CHOICE).

Par exemple :

1 - SINGLE SIDE 2 - DOUBLE SIDE

Après avoir répondu à ce choix, le message suivant sera affiché :

STRIKE A KEY WHEN READY

Dès l'enfoncement de n'importe quelle touche la point d'entrée DSKFMT - 401C sera appelé et la formatage commencera. Par contre, si vous enfoncez CTRL-STOP ou CTRL-C, le message "ABORTED" sera produit et l'on quitte la routine sans aucune action.

En entrée :

CARRY = 1 FORMATAGE sous environnement Bealé

CARRY = 0 FORMATAGE sous environnement MSX DOS
ou HL = adresse de la zone de mémoire libre,
BC = longueur de la zone de mémoire libre

En sortie :

Voir DSKFMT 401C ci-dessus.

Registres affectés : AF, BC, DE, HL, IX, IY

Exemple :

Si le programme est destiné à être intégré à un programme Basic, il est de loin plus facile de passer directement CALL FORMAT dans le programme Basic plutôt que de passer par le langage machine. Par contre, pour un programme en langage machine sous MSX-DOS, vous pouvez procéder comme suit : pour l'auteur n'importe quelle disquette. Remarque également la méthode employée pour appeler la routine en 4025H du contrôleur. Si cette méthode ne vous est pas familière, voyez le chapitre 8.

		DOS	100
0100	2A 0A 00 START:	LD HL, 1000H	adresse base du DOS
0103	11 1A 01	LD DE, PGEND	adresse fin de PRO
0106	D7	OR A	CI = 0
0107	ED 52	BBC 1A, BC	HL = lg zone libre
0109	44	D, D	longueur -> BC
010A	4D	D, D	
010B	EB	EX DE, HL	HL = adr zone libre
010C	DD 21 25 40	LD IX, 4025H	routine format
0110	FD 21 21 FD	LD IY, 10FD21H	Slot contrôleur 0
0114	CD 1C 00	CALL CALSLY	appel 1C slot IY
0117	C3 00 00	JP SLYT	Retour au Dos
011A	00 PGEND:	DEFB 0	Adresse fin PRO

Ce point d'entrée n'est présent que sur les dernières versions de contrôleurs (celles apparues avec le MSX21. Il permet de désélectionner les deux unités de disquettes de ce contrôleur et d'arrêter immédiatement leurs moteurs.

Pour rendre votre programme compatible avec tous les types de versions, testez d'abord si la position 401FH contient 00H. Si tel est le cas, n'appellez pas ce point d'entrée car il n'existe pas. Si vous persistez à l'utiliser, alors l'effet produit sera celui du point d'entrée 4022H BASIC, à savoir le retour au Basic.

En entrées Rien

En sorties Rien

Registres affectés: AF, BC, DE, HL, IX, IY

Exemples

Cet appel est utile lorsque votre programme compte travailler avec les "interruptious interdites" ou lorsqu'il supprime les "Hooks" réservés aux interruptions. Car, dans ce cas, le moteur du lecteur à partir duquel votre programme a été chargé ne s'arrêterait pas du fait que le "driver" du disque emploie un délai créé à partir des "interruptious" pour arrêter le moteur. Ce programme montre comment tester si le point d'entrée existe et, dans ce cas, arrête immédiatement les moteurs des disques A et B; autrement le programme provoque un délai suffisant pour que le moteur du disque s'arrête de lui-même.

ORG 0100

```
0100 DD 21 1F 40 START: LD 1X,401F 1 Test si 401F = 00
0104 3E 00 LD A,00 ; Si oui, délai
0106 DD DE 00 CP 41X+01 ; pour que moteur
0109 2B 09 JR 2,DELA1 ; s'arrête.
010B FD 2A 21 FB LD 1Y,(0FB211) ; Si non, arrêt
010F CD 1C 00 CALL 1C ; par appel 401F.
0112 18 0C JR FIN ;
0114 06 80 DELA1: LD B,80 ; 128 fois
0116 21 00 00 DELA11: LD HL,0 ; 65536 pour arrêt
0119 2B 0C REPEAT: DEC HL ; du moteur
011A 7C LD A,1 ; de lui-même.
011B B5 OR L ;
011D 20 FB JR NZ,REPEAT ;
011E 10 F6 DJNZ DELA11 ;
0120 F3 FIN DB ;
0121 ... ... ; suite du programme
```

Ce point d'entrée n'est présent que sur les dernières versions de contrôleurs apparues avec le MSX2. Il permet de désélectionner et d'arrêter le moteur de TOUTES les disques de votre MSX indépendamment du contrôleur auquel ille sont connectés.

Pour savoir sur quelle version tourne votre programme, testez simplement le premier octet de ce point d'entrée. S'il s'y trouve C9H, il s'agit d'une ancienne version non équipée du point d'entrée.

En entrées Rien

En sorties Rien

Registres affectés: AF, BC, DE, HL, IX, IY

Exemple:

Voir ci-dessus mais remplacez 401FH par 4029H et 00H par C9H aux endroits appropriés.

Cet applet permet de quitter un programme en langage machine et retourner avec MSX-DOS pour retourner à la routine d'initialisation de Basic. Il agit donc comme si vous allumiez votre ordinateur sans y introduire une disquette.

Trois options se présentent, selon le contenu de la position F340H.

II F340H = 00H

Si la position F340 contient 00H au moment de l'applet, le programme AUTDEXEC.BAS sera recherché sur la disquette couramment sélectionnée par le MSX-DOS.

- S'il existe, il sera chargé et exécuté.

- Dans le cas contraire, après avoir éventuellement demandé la date (si vous n'avez pas de dateur électronique, avec du MBXT1), un RUN automatique sera effectué.

Ce RUN permet le lancement d'un programme Basic qui résiderait à l'adresse donnée par le pointeur F676H (XTTABI). Le programme doit être en plusieurs segments et ne pas être en ASCII et aura été déposé au mémoire par le programme en langage machine.

Si les trois premières positions sont données par le pointeur F676H sont à 00H, alors les messages de copyright du Disk-Basic s'afficheront à l'écran et aucun programme ne sera exécuté.

2I F360H <> 00H - 0000H <> C3H

Si la position F360H contient autre chose que 00H et que la position RAM 0000H est différente de C3H alors tout un RUN automatique sera exécuté comme dans le paragraphe précédent.

3I F340H <> 00H - 0000H = C3H

Si la position F340H contient autre chose que 00H et que la position RAM 0000H est égale à C3H (environnement MSX-DOS) alors la programme Basic dont le nom est indiqué à partir de 0001H de la RAM (et dont la longueur du nom est indiquée en 0000H) sera recherchée et exécutée.

En résumé:

- 1I F360 = 0 : AUTDEXEC.DOS existe > exécution AUTDEXEC.BAS
- 2I F360 = 0 : AUTDEXEC.BAS n'existe pas -> demande la date RUN et si programme Basic en mémoire -> exécution si si pas de programme Basic -> message copyright
- 3I F340 <> 0 : RAM 0000H <> C3H -> voir a) et b) ci-dessus

4I F340 <> 0 : RAM 0000H = C3H -> recherche et exécution le programme Basic dont le nom est trouvé à partir de 0001H. Si la longueur du nom de programme indiquée par 0000H est 0 voir a) et b) ci-dessus.

En entrée:

F360H = 0 pour exécuter AUTOEXEC.BAS s'il existe. Sinon pour demander la date (MSX1) et exécuter un éventuel programme Basic se trouvant en mémoire. Si la valeur pointée par F676H vaut 00H-00H, alors s'affichent les messages de copyright du Disk-Basic.

F360H <> 0 Si on est dans l'environnement Disk-Basic, exécution d'un éventuel programme Basic se trouvant en mémoire ou affichage des messages de copyright du Disk-Basic.

2I Si on est dans l'environnement MSX-DOS, exécution du programme Basic dont le nom est donné en mémoire à l'adresse de la DTA (Disk Transfer Address, par défaut 00B0H). En cas d'absence de nom de programme dans la DTA, exécution d'un éventuel programme Basic se trouvant en mémoire ou retour au message de copyright du Disk-Basic.

00B0H Dans l'environnement MSX-DOS, placez ce 00B0H la longueur du nom du programme Basic que vous placez à partir de 00B0H.

00B0H Dans l'environnement MSX-DOS, placez ce 00B0H le nom du programme Basic à exécuter sous la forme habituelle "nnnnnnnn". Le nom du programme peut être précédé d'un caractère d'espace qui sera ignoré par la routine et qui ne doit pas être comptabilisé dans la longueur du nom du programme placé en 00B0H.

En point d'entrée est donc essentiellement utile dans l'environnement MSX-DOS pour chaîner un programme de Basic avec un programme MSX-DOS. Revenez ici après avoir consulté le chapitre 7 et 8...

Exemple 1:

Nous allons montrer ici comment appeler le programme Basic 'EXEMPLE.BAS' à partir d'un programme MSX-DOS.

```

0100      ORG 0100
...      | Corps programme
...      |
...      | MSX-DOS
...      |

04FB 3E 01      Basic: LD A,1      | F340 = pas de
04FD 32 40 E3    LD IE3401,A      | autoexec
0500 21 1A 05    LD HL,NOM      | Déplacement nom
0503 11 80 0D    LD DE,0080     | programme et
0506 01 0D 00    LD BC,0D       | longueur dans BI
0509 ED 80      LDIR            | et 0080.
050B F0 2A 21 FB LD IX,10F821)   | IX = Slot ctrl 0
050F DD 21 22 40 LD IX,4022     | IX = Adresse call
0513 C3 1C 00    JP 001C        | point d'entrée
0516 0C          DEF0 0C        | Longueur nom
0517 45 58 43 4D DEFN 'EXEMPLE.BAS' | Nom du programme
051B 50 4C 45 2E
051F 42 41 33

```

Exemple 2:

Pour quitter un programme MSX-DOS et revenir au message de copyright du Disk Basic, vous pouvez utiliser l'exemple suivant.

```

0100      ORG 0100
...      | Corps programme
...      |
...      | MSX-DOS
...      |

1000 3E 01      LD A,1
1002 32 40 F3    LD IE3401,A
1005 3E 00      LD A,0
1007 32 80 00    LD IE8000,A
100A 2A 76 F6    LD IE,10F676) | Efface 3 première
100D 06 03      LD P,3          | octets de la zone
100F 36 00      LD IRL,0       | réservée au
1011 10 FA      DJNZ REPEAT     | programme BASIC
1013 FD 2A 21 FB LD IX,10F821)   | IX = Slot ctrl 0
1017 DD 21 22 40 LD IX,4022     | IX = Adresse call
101B C3 1C 00    JP 001C        | Appel du Basic

```

Exemple 3:

Cet exemple vous montre comment quitter un programme MSX-DOS et provoquer le chargement et l'exécution du programme AUTOEXEC.BAS, s'il existe.

```

0100      ORG 0100
...      | Corps programme
...      |
...      | MSX-DOS
...      |

1000 3E 00      LD A,0
1002 32 40 F3    LD IE3401,A
1005 2A 76 FA    LD IE,10F676) | Efface 3 première
1008 06 03      LD P,3          | octets de la zone
100A 36 00      LD IRL,0       | réservée au
100D 10 FA      DJNZ REPEAT     | programme BASIC
100F FD 2A 21 FB LD IX,10F821)   | IX = Slot ctrl 0
1012 DD 21 22 40 LD IX,4022     | IX = Adresse call
1016 C3 1C 00    JP 001C        | Appel du Basic

```

Exemple 4:

Cet exemple vous montre comment installer un programme Basic sous le nom PRINT'BONJOUR';END) en mémoire et lancer son exécution à partir d'un programme MSX-DOS.

```

0100      ORG 0100
...      | Corps programme
...      |
...      | MSX-DOS
...      |

1000 3E 01      LD A,1
1002 32 40 E3    LD IE3401,A
1005 3E 00      LD A,0
1007 32 80 00    LD IE8000,A
100A ED 5B 76 F6 LD DE,10F676) | Copie programme
100D 21 21 10    LD IRL,BASIC  | en position
1011 01 13 60    LD BC,LONG    | mémoire correcte
1014 ED 00      LDIR
1016 FD 2A 21 FB LD IX,10F821)   | IX = Slot ctrl 0
101A DD 21 22 40 LD IX,4022     | IX = Adresse call
101E C3 1C 10    JP 001C        | Appel du Basic
1021 0C 80 00 00 DEF0 0C,80,0A,00 | Pointeur ligne +
1024 91          DEF0 91        | n° ligne
1025 22 42 4F 4E DEFN "BONJOUR" | Code PRINT
1029 4A 4F 55 52 DEFN "BONJOUR" | 'BONJOUR'
102D 22
102E 3A 81      DEF0 3A,81      | Code i code END
1030 00 00 00    DEF0 00,00,00 | Code fin prg
1033 13 00      LONDI DEF0 0013 | Longueur prg

```

Cette routine est pratiquement inutile pour le programmeur MSX-DOS. Elle fournit dans l'accumulateur un octet qui indique dans quel slot primaire et secondaire ce contrôleur réside sous la forme classique E00055FF. Pour atteindre ce point d'entrée, il faut d'abord sélectionner cette ROM, donc connaître le slot primaire et secondaire où il réside...

En entrée: Rien

En sortie: A = Nr du slot de ce contrôleur E00055FF

Registres affectés: AF, BC, DE, HL, IX, IY

6.12 0000 GETTOP GET TOP of user memory

Ce point d'entrée fournit dans HL la plus haute adresse RAM que l'utilisateur peut employer dans l'environnement MSX-DOS (voir pointeur F34B dans le chapitre 5.2).

En entrée: Rien

En sortie: HL = adresse mémoire maximum pour l'utilisateur

BASIC-DOS et MSX-DOS en langage-machine

Le MSX-DOS et le DISK-Basic ne vous offrent pas seulement tout ce qui a été vu dans les chapitres 3 et 4, mais également une série de routines garanties par la norme MSX qui vont permettre de manipuler le hardware de votre MSX à partir de programmes en langage machine.

Manipuler le hardware signifie disposer de routines qui permettront, par exemple, de saisir des caractères au clavier, d'afficher ou d'imprimer des zones mémoire, d'écrire, de fermer, de lire et d'écrire des fichiers, etc...

Ce chapitre va nous décrire complètement toutes les possibilités offertes et sera complété de nombreux exemples.

7.1 Pourquoi un BASIC-DOS et un MSX-DOS?

Vous savez déjà que pour utiliser le MSX-DOS, il faut disposer d'un MSX équipé de 64K de mémoire RAM et des deux fichiers MSXDOS.SYS et COMMAND.COM. Par contre, il suffit de 32K pour déjà pouvoir utiliser le DISK-BASIC sans autre logiciel que celui fourni par la ROM du contrôleur disque.

Pour permettre aux possesseurs de MSX 32K d'utiliser aussi les facilités offertes par les routines décrites dans le suite de ce chapitre, MICROSOFT a pourvu la ROM du contrôleur disque de tout ce qu'il fallait pour simuler le travail d'un MSX de 64K mais sous l'environnement du BASIC et à d. avec les deux premières pages contenant la ROM BIOS et la ROM BASIC.

Cette facilité permet aux maisons de software de développer des logiciels travaillant avec disquette sans devoir développer ces logiciels sous les deux environnements MSX-DOS et DISK-BASIC.

Le BASIC-DOS va offrir toutes les possibilités du MSX-DOS. À l'exception bien entendu des commandes vues dans le chapitre précédent. Le travail sous BASIC-DOS implique de bien connaître son environnement mémoire. Nous allons donc replacer ci-dessous le diagramme mémoire du BASIC, tant au point de vue des ROM que des RAM. Bien entendu, les slots représentés dans ce dessin peuvent changer d'une machine à l'autre ou ce qui concerne la place des RAM et même d'une configuration à l'autre ou ce qui concerne l'emplacement de la ROM du Contrôleur disque. Par contre, les deux ROM du BIOS et du BASIC sont fixes par la norme. Il n'a pas non plus à tenir compte que certains MSX1 et la majorité des MSX2 travaillent également avec des slots étendus ou slots secondaires car l'important dans ce tableau est de considérer verticalement de quoi se composent les 4K en Disk-Basic.

SLOT 0	SLOT 1	SLOT 2	SLOT 3	
+	+	+	+	FFFF
RAM				PAGE 3
+	+	+	+	C000
+	+	+	+	FFFF
RAM				PAGE 2
+	+	+	+	B000
+	+	+	+	7FFF
ROM				
BASIC	DISK			PAGE 1
+	+	+	+	4000
+	+	+	+	3FFF
ROM				
				PAGE 0
BIOS				
+	+	+	+	0000

Que voyons-nous dans ce tableau ? Essentiellement que la page 0 (0000-3FFF) est constamment occupée par la ROM BIOS, que la page 1 (4000-7FFF) est occupée soit par la ROM BASIC, soit par la ROM DISK au vu de l'instruction en cours est une instruction du BASIC normal ou du BASIC-DISK.

Cette même ROM DISK sera également chargée lorsque votre programme BASIC aura lancé soit par `LOAD`, soit par la fonction `USR`, un programme en langage machine utilisant les routines du BASIC-DOS.

Les pages 2 (8000-BFFF) et 3 (C000-FFFF) sont occupées par des RAMs qui peuvent se trouver dans n'importe quel slot primaire ou secondaire dépendant de la marque et du modèle de votre MSX.

De ce tableau, il découle qu'un programme en langage machine ne peut fatalement être implanté que dans les pages 2 et 3, car elles sont les seules à contenir de la RAM.

D'autre part, le chapitre 5 nous a expliqué qu'une partie de la page 3 était réservée pour le BASIC et pour les disques. C'est donc dans cette page 3 que MICROSOFT a implanté le point d'entrée principal du BASIC-DOS.

POINT D'ENTREE DU BASIC-DOS = F3D1

7.3 Le MSX-DOS

L'environnement associé au MSX-DOS s'apparente à celui des machines professionnelles. Dans celles-ci, l'intégrité de l'espace d'adressage du microprocesseur 1641 est occupé par les RAM, il n'y a donc plus de ROM BIOS du BASIC installés et desura dans cet espace d'adressage.

De ce fait, le système d'exploitation disque doit être chargé en mémoire RAM à l'allumage du système à partir de fichiers (MSXDOS.SYS et COMMAND.COM - Revoyez le chapitre 5 pour le détail de cette installation).

Voici donc le carte mémoire dans l'environnement MSX-DOS.

-----	FFFF
ZONE DE COMMUNICATION DU BASIC	
-----	F300
ZONE DE COMMUNICATION FIXE DES DISQUES	
-----	F1C9
ZONE VARIABLE DES TAMpons DES DISQUES	
-----	K000
ZONE FIXE CONTENANT LE SYSTEME D'EXPLOITATION DES DISQUES (MSX DOS)	Encore appelée KERNEL.
-----	Yyyy
ZONE RESERVEE AU PROGRAMME UTILISATEUR	
-----	0100
PAGE 0 DU MSX DOS	
-----	0000

Tout le problème vient, dans l'environnement MSX DOS, du fait que la zone des tampons des disques (Sector buffer, Directory buffer, Workarea buffer et les FATs) est de taille variable. En effet, celle-ci dépend du nombre de contrôleurs connectés et de la méthode d'allumage de votre MSX (lever la touche Control enfoncée ou pas).

De toute façon, cette zone s'implante sous la zone de communication fixe des disques et s'agrandit vers le bas au plus tôt y a de contrôleurs disques connectés. L'adresse F349 (zone de communication fixe des disques) contient un pointeur donnant la plus basse adresse de la zone des tampons disques (voir chapitre 5). A titre d'exemple, pour un seul disque connecté tel que dans le MSX2 V0235 de PHILIPS, l'adresse xxxx est 0F95.

Sous la zone des tampons des disques, s'implante le système d'exploitation du MSX-DOS proprement dit. Cette zone est découpée en plusieurs parties distinctes qui ont été analysées dans le chapitre 5.

La première de ces parties a été testée directement par la ROM du contrôleur disque et contient les routines de communication des disques. L'adresse F34B de la zone de communication fixe des disques fournit la plus basse adresse utilisée par ces routines de communications.

Sous cette première partie est implanté le KERNEL (Noyau) du MSX-DOS. C'est lui qui contient toutes les routines destinées au fonctionnement du MSX-DOS qui vont être décrites dans la suite de ce chapitre. Nous verrons, dans la section suivante de ce chapitre, la description de la PAGE 0 dans laquelle est indiquée la plus basse adresse utilisée par le KERNEL et qui servira de liste extrême pour notre programme utilisateur.

Enfin, sous le KERNEL, s'implantent un stack de 256 octets qui est le stack par défaut du programmeur et le code machine des commandes latérales du DOS a.s.d. Les commandes DIR, DATE, TYPE, etc. vus au chapitre précédent. Ce stack et ces commandes latérales occupent 400H positions qu'il faut donc soustraire de la plus basse adresse du KERNEL (rendre à un multiple de 256 octets pour obtenir la plus basse adresse qui est 0200H dans un MSX2 V0235 ou V0255).

Ce stack et les commandes latérales peuvent être récupérées par le programme de l'utilisateur pourvu qu'il veuille réserver un autre espace pour le stack du système et qu'il n'emploie pas ces commandes latérales.

En dehors de la page 0 (0000-0100H), tout le reste de la mémoire est disponible pour l'utilisateur.

La page 0 (0-0100H) de la mémoire est réservée au MSX-DOS et contient une série de codes d'entrée et de zones qui vont servir à exécuter toutes les fonctions offertes par le système de MSX-DOS et également à interconecter entre programme avec des routines présentes dans les ROMs BIOS et BIOS. Cette page 0 est pratiquement complétée avec celle de CP/M. Nous réviserons donc la lecture les différences entre ED/M et MSX-DOS en soulignant ceux de la liste des adresses non compatibles CP/M.

ADR. LONG NOM DESCRIPTION

0000 3 MBDOT L'adresse 0 contient une instruction de saut vers une routine qui va réinitialiser le MSX-DOS. De d'entrée, elle, et vous exécutez le jump vers cette adresse, vous programmez d'erreur et redonne la mise à l'initialisation de MSX-DOS, le contenu des adresses 1 et 2 formeront ensuite un pointeur auquel on pourra accéder et effectuer pour accéder aux routines qui seront décrites plus loin.

0003e 1 IGBYT Le 10-BYTE n'est pas supporté en MSX-DOS et nous trouvons donc 00 à cette adresse.

00041 1 DISK Le code du disque de défaut est indiqué ici en CP/M. Ce n'est pas la case en MSX-DOS. Employez le code 19 décrit plus loin pour obtenir le code de disque de défaut.

0005 3 BIOS L'adresse 5 contient une instruction de saut vers le point d'entrée principal de MSX-DOS pour charger les routines qui vont être décrites dans la suite de ce chapitre. Les codes des adresses 6 et 7 sont également réservés à des bases de données du KERNEL de MSX-DOS. On peut donc utiliser le mot présent en 0006-0007 pour connaître la plus haute adresse de la RAM que vous pouvez utiliser pour votre programme en découlant simplement à du code de 0006-0007.

000E 3 RPSLT L'adresse 0E contient une instruction de saut vers une routine qui permet de lire dans l'accumulateur, l'adresse mémoire HL de la liste dont le caractère de sélection se trouve dans l'accumulateur.

0010 3 WRBI L'adresse 10 contient une instruction de saut vers une routine qui permet d'écrire le contenu du registre E vers l'adresse mémoire HL de la liste dont le caractère de sélection se trouve dans l'accumulateur.

001D 3 CASIT L'adresse 1D contient une instruction de saut vers une routine qui permet d'appeler une routine à l'adresse 1E du jeu dont le caractère de sélection se trouve dans IV.

00241 3 ENAB1 L'adresse 24 contient une instruction de saut vers une routine qui permet d'activer le jeu de données de la liste dont le caractère de sélection se trouve dans l'accumulateur et dont l'adresse figure dans le registre HL (voir chapitre 9).

002B 3 RST26 Les programmes Debugger comme ISD - DDT utilisent certainement le RST26 comme break point. Cependant, le MSX-DOS utilise la liste 26 qui est utilisée par la routine interrompue. Ces programmes sont donc pechés pour utiliser le RST 26.

0030 3 CALLF L'adresse 30 contient une instruction de saut vers une routine qui permet d'appeler une routine suivie de la procédure suivante (voir chapitre 9).

RST 30
DEFB 0101 de sélection
DEFB Adresse de la routine

003B 3 INTPT L'adresse 3B contient une instruction de saut vers la routine d'interruption d'entrée du ROM BIOS en MSX-DOS. En CP/M, cette routine contient en général une instruction de saut vers la routine de la routine Debugger comme DDT ou Z80. Ce programme doit ou doit avoir dans la liste de saut pour sauter au MSX-DOS.

003D 11 Les 11 octets présents à partir de l'adresse 3d forment une routine décrite par le MSX-DOS pour charger les données secondaires de la page 3 (0000-FFFF) et dans les entrées suivantes:

A : Code des entrées principales à sélectionner.
d : Code des entrées principales à sélectionner en retour de routine.
H : Numéro des entrées secondaires à conserver.
D : Code des entrées secondaires à sélectionner.

En outre, le registre L contient le contenu précédent de registre de sélection des entrées secondaires.

0043 5 Les 5 octets de l'adresse 43 contiennent

Les quatre gènes du locus requièrent des sites secondaires, la région A doit contenir la sous-séquence de sélection dans la région de sélection des sites primaires. La région L doit contenir la séquence de sélection dans la région de sélection des sites secondaires. La région B doit contenir le code de sélection et se trouve dans la région de sélection des sites primaires ou celui de la région.

00421 10

Les 10 Delta de cette section ont la même mission que celle de l'adresse 004 à l'exception que le code à lancer dans le registre en sélection des bits selon des bits 17 se place dans la sortie 6.

0050 14 FEB 1964

Les le hyres de l'adresses 50 continuent le premier argument d'un second appel de programme MEN 508. On appelle premier argument le liste continue qui null la non du programme après un espace au moins de séparation. Le second argument doit être espace du premier pas moins un espace, ainsi, si vous appelez COMPTON, BRETEL,ASC d'ELITE,CUP, vous recevrez la notice du disque de du livraier après un SE, suivi du nom de livraier entre 3 caractères et de l'extension du livraier entre 3 caractères. Ces labels doivent directement pour le PCB de la fin.

094E 1A 1E02

Les id entifie de l'adresse 49 sont
steerv de ou essant argumet e'une
comende au d'un programme MS-DOS
suivie le s'gle de l'ite el-deux
pau la d'ier argumet, l'xi.CDF est
le s'one argumet dans l'exemple
el-deux.

0000 120 01A

Les 128 octets de sortie de l'adresse 80 forment le tampon DMA 1024 octets adressé adresses du DMA 1024 octets adressé par défaut de système. Lorsqu'on lire un enregistrement d'un fichier, le contenu de son enregistrement es placé en mémoire de l'adresse DMA. Par défaut le système place ce tampon d'adresse 80.

0100 0000 1111

d'après la carte adresse et jusqu'à l'acte eventuel d'adresse donnée par les géologues et c. et se trouve que l'on appelle le PPA ou TRANSITION PROGRAM AREA est qui est la limite de la ZONE DES PROGRAMES TRANSITIONAIRES. C'est en fait l'espace admissible réservé pour installer vos propres programmes. Cet espace dépend du nombre de contributeurs

disques de votre exécutif et de la méthode d'ajustement de votre M51. A l'issue de l'essai, vous obtiendrez de DICO 4 D505 pour les machines de la gamme M512 de PHILIPS avec D505 et de 56325 de la librairie pour le programmeur M512-005. Ce nombre correspond donc à pas près au double de la quantité d'un M512 sans disque.

Consultez le tableau. Le premier disque est un simple jeu de données, pour servir de programme MSX-DOS de valeur ajoutée. Il suffit de programmer un seul et l'adresse D. Vous recevrez ainsi de façon progressive l'indicateur P2 de MSX-DOS.

ADRESSE POUR SORTIR DU MEX-DO - 0

Deuxième donnée très supérieure, c'est l'absence de point d'entrée principal du MSX DOM, bien que cette donnée soit en soi vraiment laide d'aspect.

```

1 POINT 8 ENIREC MAX DOB = 0005

```

prolaine liasse; l'adresse de mise à la pisse, hase que vous
passez utiliser par votre propre, et donnez
indirectement que le point d'entrée principal du M1-000
suis-je l'outil de calcul et de la présence de et et l'
jour nouvelle celle pisse hase adresse. Cependant, parce
de derrier un adresse suffisant pour le pisse de éviter
l'ATTACK. Il est d'ailleurs recommandé de pisse celle pisse
Jetez dans le M1-000 M1-000 dans de l'adresse d'origine par le
passez l'adresse d'origine d'7, pour passer à l'adresse une
sile de 256 octets et fixer le "IP" de votre adresse,
passez dans l'outil.

```

| LD  R6, 10000H |
| IO  SP, R6 |
| DEC R6 |
| LD  ITOP, R6 |

```

Les unités d'enseignement de l'initiation de programmation, toutes les semaines de consultation de la salle de travail sont assurées dans le cadre d'un cours de programmation de la ROM. Les unités de programmation de la ROM sont assurées dans le cadre d'un cours de programmation de la ROM. Les unités de programmation de la ROM sont assurées dans le cadre d'un cours de programmation de la ROM.

Les routines de conseil en des clin secondaires gérées à partir de l'adresse 38 nd sont normalement pas évaluées par les programmes utilisateur nait appel des utilisateurs par les routines et attend de conseil en de clot 1 (BGL) - MRSU - CALSI - ENABT - CALFI. - Ju l'us al accipienda

uniquement pour le tableau de la page 0 soit compilé à
pour que vous évitiez d'utiliser ces zones réservées.

Autre élément, très important quand on programme en MSX-DOS,
est la possibilité d'appeler un programme en y adjoignant
jusqu'à deux arguments. Si, par exemple, vous réalisez un
programme en langage machine qui convertisse un fichier de
texte écrit avec le code MSX en un fichier de texte en code
ISO-7bit où les caractères accentués du français sont
différents, vous pouvez directement à l'appel du programme
spécifier quel fichier a été à traduire et en quel autre nom
de fichier vous voulez que le résultat soit sauve. Si
CONVERT.COM est le nom de votre programme, vous pouvez
l'appeler en posant par exemple :

CONVERT B:TEXT.MSX TEXT.ISO

Dans cet exemple B:TEXT.MSX est le premier "argument" et
TEXT.ISO est le second. Le système d'exploitation MSX-DOS va
ici nous aider considérablement en plaçant lui-même le
premier argument à l'adresse 5C et le second à l'adresse 6C.
Il ne se contente pas seulement de les y placer, mais il va
aussi les formater au FCB c.à.d. convertir le nom de disque
par défaut ou non en son numéro, et aligner le nom du
fichier et son extension suivant le format du FCB. Ainsi,
pour l'exemple ci-dessus nous trouverons en 5C et en 6C :

```

1 - Numéro du disque
|
5C D2 54 45 5B 54 20-20-20-20-40 53 5B 00 00 00 00
   T E X T           M S X

+ - Disque par défaut
|
6C 00 54 45 5B 54 20-20-20-20-40 53 4F 00 00 00 00
   T E X T           I S O
  
```

En 5C, le code 02 signifie que le fichier TEXT.MSX est à
rechercher sur le disque B; et en 6C le code 00 signifie
qu'il faut générer le fichier TEXT.ISO sur le disque par
défaut. Pour ouvrir ces fichiers, il nous suffira de copier
les 16 octets de 5C vers l'adresse où nous comptons établir
le FCB de l'input de ce fichier et de même pour le second
argument. N'oubliez pas, au fait, que le FCB occupe 37
octets et que dès lors il ne dispose pas d'espace de place en
5C ou 6C.

En Basic, chaque lecture de fichier se fait dans une
variable fixée par INPUT# pour les fichiers séquentiels ou
par FIELD# pour les fichiers à accès direct. En MSX-DOS, les
lectures et écritures se font par référence à une zone de
mémoire que l'on appelle DMA BUFFER en CP/M ou DTA (Disk
Transfer Address) en MSX-DOS. L'adresse de ce tampon pourra
être fixée au gré du programmeur par une fonction spéciale
que nous verrons plus loin. Cependant, si le programmeur
omet de fixer cette adresse, le système pourvoit une zone
par défaut où les lectures et écritures d'un fichier seront
transférées : c'est le Default Disk Transfer Address (DDTA)
présent à l'adresse 00H de la page 0.

Finalement, quelques mots à propos de la TPA où zone où
s'installent vos propres programmes. Sachez d'abord que la

simple fait de poser le nom d'un programme MSX-DOS après
l'indicateur A: du MSX-DOS provoque le chargement de ce
programme à partir de l'adresse 0100H. Il n'y a donc pas
d'entrée dans un programme MSX-DOS spécialement qu'il doit
être chargé de telle adresse à telle autre adresse et
débarquer à une troisième adresse comme dans les programmes
binaires du Basic chargés par BASIC. Le chargement se fera
toujours en 0100H. A propos du nom d'un programme MSX-DOS,
rappelons ici que le nom du programme doit contenir
l'extension ".COM" mais qu'il n'est pas nécessaire de taper
cette extension pour appeler le programme.

Les fonctions qui vont être décrites dans cette section sont valables aussi bien en BASIC-DOU qu'en MOX-DOU. Le lien employé pour ces fonctions en Anglais est "System Call" ou Appel System mais pour le français, nous écrirons de les appeler Fonctions.

L'emploi des fonctions avec le langage machine est le même et correspond à celui de CP/M ou d'assemblé DOS sous le MS-DOS par exemple.

Chaque fonction a reçu sa propre, doit appeler une fonction, il suffit de dire dans le registre C du microprocesseur le numéro de la fonction désirée et appeler le point d'entrée principal de MSX-DOS (0005H) ou le point d'entrée principal du BIOS C-005 (F37DH) dépendant de l'environnement où l'on se trouve.

Dans le reste de cette cellule, les exemples d'espèces furent
 tantôt rattachés à l'environnement BNCIC BNC ou à
 l'environnement MS-DBS (selon), indiquant que la prise de
 décision était bien aux deux environnements sans un
 équilibre plus attendu de la part d'environnement ou F3D1 si
 bien selon l'implication à l'adresse même de celle-ci.

Bien entendu, pour certaines fonctions, il ne s'agit pas de
 enlever le poids de la fonction dans le registre C ou
 d'appeler le point d'entrée de MSX-DOS. Il faudra aussi lui
 fournir des paramètres, sous par exemple quel caractère
 afficher, l'heure pour une fonction d'affichage ou quel
 caractère pour une fonction de lecture. Les paramètres
 les paramètres d'entrée. De plus, certaines fonctions vont
 nous rendre illégal qui sera attaché avec des registres
 de superprocesseur ou en mémoire. Ce sont les paramètres de

Deus é ille dos lionelano qe ve leivre, voel trouverse
ligeasse un indicio de compatibilidde ivce i CP/M. En
efei, o NGK-DOS ee praeqe compatielme compatie ave
ie CP/M; ehaque lionelln sera dnt ressignate joem
compatie bu non ave i CP/M. D'seie peri, e ille dos
realetre orieivru ser ie lionelln sei auti donne.

7.5.1 FUNCTION ON SYSTEM RESET

Dans l'environnement MSX-DOS, cela fonctionnait jusqu'à ce que
règle qu'il se soit à l'adresse 0000H. Cela signifie que le
programme en cours s'arrête et que l'on reviens à
l'indifférence du MSX-BIOS.

Dans l'environnement BASIC-DOS, cette fonction provoque l'arrêt du programme et le retour à l'invite de BASIC.

En entrée	Rien
En sortie	Rien
Compatible	Oui

Exemple : Pour lancer le programme, vous avez le choix d'utiliser soit l'icône ou, en MSX-DOS, un saut à l'adresse 0000.

Ld	C, 0	OU	2P	00001
JP	0037TH			

7.3.2 FUNCTION OF GENETIC UNITS

La fonction permet de sélectionner une caractéristique quel que soit le niveau. La caractéristique sélectionnée à l'interne est la routine vérifiée logiquement, et elle peut être l'interface C, ou peut être la fonction CO est sélectionnée également, ou le Control-P est sélectionné tout ce qui sera affiché sera aussi dirigé vers l'imprimante, ou Control-M et alors l'interface vers l'imprimante sera supprimée. Attention ces caractéristiques graphiques à deux codes CO-XXI, car ce seul code est relatif.

En entrée : Rée
 En sortie : Les registres A et L contiennent le code de
 l'instruction passée en entrée.
 Préservés : Les registres C, P et E sont préservés.
 Copiés : Del

Exemple : Voici notre premier programme DOS. Il se fait qu'attirer l'opérateur à taper une lettre à la fois. Si la lettre est la lettre 'a' ou 'A', alors le programme s'arrête et l'on revient à l'invite de commande.

496 0000

```

0000 0E04      START:  I B  C, 1      |  A6=INDUTY(1)
0002 CD7DF3   CALL  OF37D      |
0005 FE3A     CP      '1'      |
0007 20F7     JR      NZ, START |
0009 0E00     LD      C, 0      |
000B C37DF3   JP      OF37D      |  Relout: au

```

J'aimerais aussi de notre premier exemple, nous allons
 essayer d'introduire ce petit programme dans le
 fichier ".BIN" qui doit être chargé plus tard par la
 commande "LOAD" (voir plus tard).

10 LEARN 200-444EFF

```

20 FOR I=H4B000 TO H4B008
30 READ A#
40 POKE I,VAL("H#"+A#)
50 NEXT I
60 BSAVE "INPUT,BIN",H4B000,H4B008,H4B008
70 END
80 DATA 0E,01,CD,7D,F3,FE,3A
90 DATA 2D,F7,DE,00,C3,7D,F3

```

Si vous désirez travailler dans l'environnement MSX-DOS et que vous ne disposez pas d'un MACRO-ASSEMBLEUR comme DEVPAD ou MACRO-80, voici le moyen de créer le fichier 'INPUT.COM' en BASIC.

```

10 OPEN "INPUT.COM" AS #1 LEN=1
20 FIELD #1, 1 AS A#
30 I=1
40 READ A#
50 IF A#="" THEN CLOSE#1
60 LET A#=CHR$(VAL("H#"+A#))
70 PUT #1,I
80 I=I+1:GOTO40
90 DATA 0E,01,CD,7D,F3,FE,3A
100 DATA 2D,F7,DE,00,C3,7D,F3,

```

Voilà, il vous suffit d'appeler le programme 'INPUT' lorsque vous aurez rechargé votre système en MSX-DOS.

7.3.3 FONCTION 02 CONSOLE OUTPUT

Cette fonction permet d'envoyer un caractère vers l'écran et l'emplacement courant du curseur. Cette fonction teste également le clavier pour vérifier si un des codes CTRL-C, CTRL-P, CTRL-N ou CTRL-S a été généré, auquel cas la fonction y associe une action.

En entrée : Le registre E doit contenir le caractère à afficher. Ce peut être un caractère typographique ou un des codes de fonction comme 07 (Beeep), 0C (lecl), etc.

En sortie : Le registre E d'origine est copié vers les registres A et L.

Préserve : Les registres C, D et E sont préservés.

Compatibilité : Oui

Exemple : Ce petit programme va simplement afficher l'écran en envoyant le code 0C, 01, vous l'appeler 'GTS.COM', vous pourrez afficher l'écran dans l'environnement MSX-DOS comme vous avez l'habitude de le faire en BASIC e.e.d. en posant simplement CLS.

ORG D100

```

0100 1C0C  START: LD  C,0C      | Code 0C = CLS
0102 0E02  LD  D,02      | Envoi vers l'écran
0104 D00500 CALL 0005      |
0107 D30000 JF  0000      | Retour au DOS

```

7.3.4 FONCTION 03 AUXILIARY INPUT

Cette fonction permet de saisir un octet en provenance de l'entrée auxiliaire. Cette entrée peut aussi être en général réservée pour l'interface RS-232-C qui permet de mettre en liaison deux ordinateurs MSX via un câble ou encore via adams et le réseau téléphonique. Cette fonction teste également les codes CTRL-C, CTRL-P et CTRL-N en provenance du clavier et le cas échéant exécute la fonction y associée.

En entrée : Rien

En sortie : Les registres A et L contiennent le caractère en provenance de l'entrée auxiliaire. Si cette entrée n'existe pas sur votre MSX, le système renvoie le code 1A qui est le code de fin de fichier.

Préserve : Les registres C, D et E sont préservés.

Compatibilité : Oui

Exemple : Ce programme affiche à l'écran tous les codes reçus de l'entrée auxiliaire jusqu'à ce qu'un code de fin de fichier (1A) arrive, auquel cas, le programme revient à l'indicateur A du DOS.

```

ORG D100
0100 0E03  START: LD  C,03      | Lire PORT AUX.
0102 D00500 CALL 0005      |
0104 FC1A  CP  1A          | Si fin fichier
0107 D00000 JF  7,0000      | retour au DOS
010A 3F    JP  E,0         | Envoi du code
010B 0E02  LD  C,02      | à l'écran
010F D00500 CALL 0005      |
0110 1B0D  JR  START      | recommence

```

7.3.5 FONCTION 04 AUXILIARY OUTPUT

La fonction 04 envoie le contenu du registre C vers l'entrée auxiliaire qui est généralement employée avec l'interface RS-232-C. Si cet interface n'est pas installé, cette fonction est sans effet. Cette fonction teste également si le clavier a généré un des codes CTRL-C, CTRL-P ou CTRL-N, auquel cas la fonction y associe une action.

En input : Le registre C doit contenir le code à envoyer à la sortie auxiliaire.

En sortie : Le registre E d'origine est copié vers les registres A et L.

Préserve : Les registres C, D et E sont préservés.

Compatibilité : Oui

Exemple : Ce programme envoie le message "Bien reçu votre message" suivi d'un marque de fin de fichier vers le port de sortie auxiliaire.

```

0100 211301  START:  ORG 0100
0103 5E      RPT:   LD  R1,MSB      | Pointeur message
0104 25      PUSH HL | Lit un code
0105 0E04    LD  E,04      | Sauve pointeur
0107 0D0500  CALL 0005    | Envoie code vers
010A 01      POP HL       | port AUX.
010B 23      INC HL       | Rappel pointeur
010C FE14    CP  14       | Incrémente le
010E 20F3    JR  NZ,APT   | Fin de message ?
0110 C30000  JP  0000     | Oui -> APT
0113 4204544 MES: DEFB 'BIEN RECU'
0117 20524B03
0118 55
011C 20544F54 DEFB 'VOTRE '
0120 8205
0122 4B455353 DEFB 'MESSAGE'
0126 114705
0129 1A      DEFB 16      | Code de libération

```

7.5.4 FONCTION 05 LIST OUTPUT

La fonction 05 envoie le contenu du registre E vers l'imprimante. Si l'imprimante n'est pas connectée ou si elle est OFF-LINE, le routine attend que vous le puissiez ON-LINE. Espérons, il est possible d'arrêter le programme et donc l'impression et vous entendez CTRL-C ou CTRL-GIMP. N'oubliez pas d'individuellement convertir la code HEX dans le code de l'imprimante et vous avez été imprimés en été.

En entrée : Le registre E doit contenir le code à imprimer.
 Le sortie : Le registre E sera copié vers les registres A et L.
 Préserve : Les registres C, D et E seront préservés.
 Compatible : Oui

Exemple : Nous allons taper l'argument qui suit le nom du programme. Rappelons qu'un MSB DOS on peut faire suivre le nom d'un programme et deux arguments et qu'ils seront placés en adresse des adresses 5C et 60. Ici, si nous appelons ce petit programme LPRINT.COM, vous obtiendrez l'impression du BONDUR et vous lisez après l'indication de LPRINT Bonjour. Attention que l'argument doit suivre les règles propres aux codes de libération.

```

0100 213000  START:  ORG 0100
0103 0600    LD  HL,0500 | écriture argument
0105 05      RPT:   LD  R1,R      | Boucle 8 fois
0106 25      PUSH BC      | Sauve compt boucle
0107 5E      PUSH HL      | Sauve pointeur
0108 0E04    LD  E,04      | Lit un code
010A 0D0500  CALL 0005    | Envoie vers impr.
010B 01      POP HL       | Rappel pointeur
010C 23      INC HL       | Incrémente le
010E 01      POP BC      | Rappel compt boucle
0110 10F3    DJNZ APT     | Répète boucle
0112 C30000  JP  0000     | Retour au DOS

```

7.5.7 FONCTION 04 DIRECT CONTROL INPUT/OUTPUT

- Si la registre E contient FF, alors il s'agit de la fonction Direct Input. Cette fonction permet d'obtenir le code qui serait prêt dans le temps du clavier et le retourner dans la registre E. Si aucun code n'a été passé, alors la registre E contiendra 00. Attention qu'il s'agit d'une seule accélération du clavier sans attente de l'appui d'un caractère et dès lors la code sera du 21rs (frappé avant l'appui de cette fonction).
- Si la registre E contient un code différent de FF, alors il s'agit de la fonction Direct Output. Elle permet d'effectuer à l'écran la code contenu dans le registre E.
- On parle de 'Direct' Input ou Output en ce sens que cette fonction évite les contrôles et dispositifs matériels de CP/M et évite qu'il n'y a été de test des codes CTRL-P, CTRL-N, CTRL-C ou CTRL-B, et s'ajoute des caractères tels que input, ni de l'ajoutage des caractères envoyés à l'écran en output.

En entrée : Le registre E doit contenir FF pour ce Direct Input et la code à émettre en Direct Output.

En sortie : En Direct Input, les registres E et L contiennent 00 et aucun code n'est prêt dans le temps du clavier ou le code de la touche dans le cas contraire.

En Direct Output, les registres A et L contiennent le code envoyé à l'écran.

Préserve : Les registres C, D et E sont préservés.
 Compatible : Oui

Exemple : L'appel de cette fonction s'est par conséquent par Digital Research qui est le concepteur de CP/M parce qu'elle est directement liée à la liste et émettent la destination des inputs/outputs. Note d'ailleurs la seule fonction qui pouvait écrire des caractères dans les écrivains à l'écran. Le MSB DOS a codé cette fonction par les deux adresses D7 et 06. Cependant, elle conserve son utilité, car c'est la seule fonction qui écrit le clavier sans attendre la frappe d'une touche. Nous allons modifier ce programme et émettre la touche ESCAPE sans attendre qu'elle le fasse. La touche ESCAPE sert de sortie du programme.

```

0100 1EFF    START:  ORG 0100
0102 0E04    LD  E,04      | fonction INPUT
0104 0D0500  CALL 0005    |
0107 5F      LD  A,0A      |
0108 0D0500  CALL 0005    | fonction OUTPUT
010B FE14    CP  14       |
010C 20F3    JR  NZ,START | Check ESCAPE
010E 20F1    JR  0,START  | Non -> START
0110 C30000  JP  0000     | Oui -> Retour DOS

```

La fonction 07 permet d'attendre le frappé d'un caractère ou plusieurs caractères sans afficher le caractère frappé et sans tester les touches CTRL-C, CTRL-P, CTRL-N ou CTRL-G.

En entrée : Rien
En sortie : Le registre A contient le code de la touche frappée.
Observer : Les registres C, D et E sont préservés.
Compatibilité : Non. La fonction 07 en CP/M s'appelle GET ID BYTE. Cette fonction CP/M n'est pas implémentée en MSX-DOS. Cependant, vu le fait que cette fonction CP/M est employée quasiment toujours, il est peu probable qu'un programme CP/M soit incompatible avec la MSX de ce site.

Exemple : Lorsqu'on crée quelques programmes comme son emploi par une personne non autorisée, il est usuel d'employer la technique où on se passe. Cependant, il est important lors de la mise au point de ce code de passer que des yeux indiscrets ne puissent voir le programme qui vient d'être écrit. Voici un programme qui demande un mot de passe (MSX) avant d'afficher l'écran.

```
0100 0E09      START: LD      C,1          ; Attache de
0102 113701    LD      DE,REG          ; "MOT DE PASSE"
0104 000500    CALL    0005          ;
0106 0405      LD      B,3          ; Boucle pour lire
0108 213101    LD      H,K,BUF        ; 3 caractères
010C 05        RPT: PUSH    BC        ; Sauve comp. boucle
010E 05        PUSH    HL          ; Sauve pointeur
0110 0E07      LD      C,07          ; Lire un code sans
0112 000500    CALL    0005          ; Echo à l'écran
0114 61        POP      HL          ; Appel pointeur
0116 77        LD      HL,(HL),A     ; Sauve code
0118 23        INC      HL          ; Inc. pointeur
011A 01        POP      BC          ; Rappel comp. boucle
011C 10F3      DJNZ    RPT          ; Répète 3 fois
011E 0403      LD      B,3          ; Compare mot de
0120 113101    LD      DE,PASS       ; passe avec
0122 7B        CHK: DEC      HL      ; "MSX"
0124 1A        LD      A,1041        ;
0126 BE        CP      HL          ;
0128 2000      JO      NZ,START      ; SI <> -> START
012A 13        INC      DE          ;
012C 10F3      DJNZ    CHK          ;
012E 1E0C      LD      C,0C          ; SI = envoi CLS
0130 0E02      LD      C,02          ;
0132 000500    CALL    0005          ;
0134 C30000    JP      0000          ; Retour au DOS
0136 000000    BUF: DEF      3      ; Tampon lecture
0138 50534D    PASS: DEF      "XSM" ; Password levert
013A 000A      MES: DEF      0D,0A  ; Message initial
013C 10F5120   DEF      "MOT "     ;
013E 113520    DEF      "DE "      ;
0140 5015353   DEF      "PASSE",F  ;
0142 453A
```

La fonction 08 attend le frappé d'un caractère ou plusieurs caractères sans afficher le caractère frappé et vérifie si un des codes CTRL-C, CTRL-P ou CTRL-N a été testé auquel cas la fonction y associe une action.

En entrée : Rien
En sortie : Le registre A contient le code du caractère qui a été frappé.
Observer : Les registres C, D et E sont préservés.
Compatibilité : Non. La fonction 08 en CP/M s'appelle GET ID BYTE. Cette fonction CP/M n'est pas implémentée en MSX-DOS. Cependant, vu le fait que cette fonction CP/M est employée quasiment toujours, il est peu probable qu'un programme CP/M soit incompatible avec la MSX pour cette fonction.

Exemple : Vous pouvez modifier l'exemple de la fonction précédente en remplaçant le code 07 à l'adresse 110H par 08. Le seul changement de fonctionnalité de ce nouveau programme sera qu'il autorise de stopper le programme par CTRL-C ou tout autre mot de passe tel demandé.

7.5.10 FONCTION 09 STRING OUTPUT

La fonction 09 permet d'afficher à l'écran une chaîne de caractères. Cette chaîne peut être constituée de caractères ou de codes MSX, y compris les codes de fonction ou les séquences ESCape. Cependant, elle doit se terminer par le caractère à l'code 2FH qui marque l'code de fin de la chaîne et doit se faire au moins une fois. Si vous devez plusieurs chaînes de codes, alors utilisez la fonction 02. Il n'y a pas de limite de longueur à la chaîne comme en BASIC. Cette fonction teste également si un des codes CTRL-C, CTRL-P, CTRL-N ou CTRL-G a été testé auquel cas il s'élève et la chaîne transmise à la fonction y associe.

En entrée : Le registre de 10 bits DE doit contenir l'adresse absolue où commencer la chaîne. Ne pas oublier de localiser la chaîne par le code 2FH. Le registre 0E contient l'adresse du code à lire.
Observer : Seul le registre C est préservé.
Compatibilité : Oui

Exemple : Nous allons écrire comment afficher l'écran, afficher un titre au milieu de la ligne, le souligner et entrer le curseur en mode souligné sur la division ligne si tout cela en un seul appel de la fonction. Voici son équivalent BASIC


```

C000 1129C0 ST447: ORG C000
C003 3628 LE DE,MAX r DE = Adresse de MAX
C008 12 LD A,20 ; Mettre 40 dans MAX
C00A 0E04 LD I,DEI,A ;
C00B C07DF3 CALL OF37D ; INPUT A=
C00B 13 JNC DE ;
C00C 1A LD A,DEI ; A = nbr car posés
C00D 13 JNC DE ;
C00E 83 ADD A,E ; A est augmenté de 1
C00F 5F LD E,A ; adresse des données
C010 1001 JR NC,NEX1 ; si fin du langage
C012 14 JNC B ;
C013 3E24 NEXT: LD A,'4' ; Mettre 4 a la fin
C015 12 LD I,DEI,A ; de la chaîne
C016 1128C0 LD DE,BUF ; PRINT A=
C019 0E09 LD C,09 ;
C01D C07DF3 CALL OF37D ;
C01E 112901 LD DE,CRLP ; Affiche CR-LP
C021 0E09 LD C,09 ;
C023 C07DF3 CALL OF37D ;
C026 C300C0 JP DIAR1 ; Reconnaître
C029 00 MAXr DEFB 0 ; nbr car. à lire
C02A 00 CNTr DEFO 0 ; nbr car. lus
C02D 00 BUFR DEFB OFR ; langage de 255 pos.
C12A 0D0A24 CRLP: DEFO 0D,0A,24 ; codes CR-LF-4

```

Les instructions de C003 et C005 permettent de fixer le nombre de caractères maximaux demandés. Les instructions de C008 à C015 servent à placer le code à dans le Buffer juste après le dernier code posé en se servant de I,DEI qui indique le nombre de caractères qui ont été posés. C016 qui indique la chaîne et C021 à C023 provoque le retour à la ligne.

7.d.12 FONCTION 00 DEL CONSOLE STATUS

La fonction 00 examine l'état du clavier. Si un caractère a été posé, alors le registre A contient FF sinon il contient 00. Pour obtenir le code du caractère posé, employez la fonction 01. Cette fonction teste aussi les codes CRLR-R, CRLR-P, CRLR-N et CRLR-C.

En entrée : Rien
 En sortie : Les registres A et L seront à 00 et le flag Z à 1 si aucun code n'a été posé. Les registres A et L contiendront FF si le flag Z sera à 0 dans le cas contraire.
 Préserve : Les registres C, D et E sont préservés.
 Compatible : Oui
 Exemple : Le programme suivant qui équivaut à la routine Basic suivante (vous pouvez l'arrêter par CRLR-C).

```

10 A=INKEY$
20 IF A="" GOTO 10
30 PRINT A$
40 GOTO 10

```

```

ORG 0100
0100 0E0B TEST: LE E,0B ; Test clavier
0102 C0D050 OR A,0000 ;
0105 87 CRR A ; A = 00 ?
0106 20F8 OR A,Z,TEST ; Oui -> TOUT
0108 0E01 JRC,01 ; Non -> 1111e
010A C0D050 CALL 0005 ; code
010D 5F PRINT: LE E,A ; Affiche le
010E 0E02 LD E,02 ; code
0110 C0D050 CALL 0005 ;
0113 C300C0 GOTO: JP TEST ; Reconnaître

```

7.5.13 FONCTION 00 GET VERSION NUMBER

Cette fonction s'appareille dans le NSX DOS que par compatibilité avec le CP/M. Elle permet, ce CP/M, de reconnaître le version de système utilisée. Comme le NSX-DOS est compatible avec le version 2.2 du CP/M, cette fonction placée 00 dans le registre H et Z dans le registre L. Cependant, elle n'est pas prévue pour réaliser des versions différentes du NSX DOS.

En entrée : Rien
 En sortie : Le registre H indique 00.
 Le registre L indique 231.
 Préserve : Les registres C, D et E sont préservés.
 Compatible : Oui, mais n'est exclusivement que CP/M.

7.5.14 FONCTION 00 DISK RESET

La fonction 00 sélectionne le disque A comme disque par défaut, place la CTR ou DMA à l'adresse par défaut 100B0H, écrit le langage secteur sur le disque s'il s'avère modifié et ré-écrit les FAIs de tous les disques et elles apparaissent modifiées. Attention pour le travail en BASIC-DOS, la valeur par défaut de DMA buffer 100B0H ne convient pas et doit donc être modifiée par le langage IA à partir de 8000H.

En entrée : Rien
 En sortie : Les registres A et L contiennent 00. Les registres DE et IX contiennent l'adresse du DPR (Disk Parameter Block) de dernier disque de votre système.
 Préserve : Seul le registre C n'est pas modifié.
 Compatible : Oui

Exemple : Il est recommandé d'employer cette fonction lors des programmes qui nécessitent des permutations de disquettes afin d'éviter d'écrire des secteurs sur le mauvais disque. De même, lorsqu'un programme s'effectue sur un autre disque par défaut, il est souhaitable d'employer cette fonction avant de quitter le programme pour rétablir le système comme à l'origine.


```

0300 0C00      START:  LD C,00      ; Reset des disques
0302 0D0500    CALL 0005      ;
0305 0E09      LD C,09        ; Alliege message d'
0307 111003    LD DE,HE5      ; Echange disques
030A 0D0500    CALL 0005      ;
030D 022103    JB SUITE       ; Suite programme
0310 1505B111  MVI DCRB 'CHANGEZ?'
0311 1E071A5A  DB 0
0313 1C453120  DEB 'LES '
031C 11455331  DEB 'DISQUES.'
0320 5545532E
0324 ....     SUITE:  ...

```

7.5.15 FONCTION DE SELECT DEFAULT DISK

Cette fonction permet de désigner le disque par défaut. En CP/M et en RM-DOS, les fonctions manipulant les disques font référence à un ordre de disque glissé qu'on nomme des disques 0 à 11. Ces ordres vont de 0 à 9 soit neuf possibilités. Le disque 0 est rattaché par 1, le B par 2 et ainsi de suite jusqu'à B pour le disque M. Le numéro 0 désigne le disque par défaut. Cette fonction permet donc de sélectionner quel sera ce disque par défaut. Cette fonction permet aussi de sélectionner le nombre de disque logique dont vous disposez sur le système.

En entrée : La table B doit indiquer quel disque sera désormais le disque par défaut suivant le table que voici: A=0 B=1 C=2 D=3 E=4 F=5 G=6 H=7 I=8 J=9 K=10 L=11. On peut aussi donner une valeur à la table B pour le disque par défaut. On peut aussi donner une valeur à la table B pour le disque par défaut.

En sortie : La registre A renvoie le nombre de disques logiques connectés au système. A=0, si vous avez un seul disque physique et que vous avez démarré l'ordinateur sans avoir appuyé sur CTR, la registre A contiendra 02 car dans ce cas vous n'avez qu'un disque. Si vous avez en parallèle les disques 0 et 1, la registre A contiendra 02 car vous avez deux disques. Si vous avez en parallèle les disques 0 et 1, la registre A contiendra 02 car vous avez deux disques. Si vous avez en parallèle les disques 0 et 1, la registre A contiendra 02 car vous avez deux disques.

Prérogative : Les registres C, D et B sont préservés.
Compatibilité : Oui, excepté l'ordre CP/M le nombre de disques n'est pas indiqué dans la registre A en sortie.

Exemple : Si vous appelez ce sous-programme qui sélectionne le disque B comme disque par défaut et si vous avez installé l'installateur de MSX-DOS vous verrez que l'installateur ne sera plus de mais B et ainsi tout programme appelé sera exécuté sur le disque B.

```

0100 1E01      START:  LD C,01      ; Disque B
0102 0E0E      LD C,0E        ; Fonction DE
0104 0D0500    CALL 0005      ; Appel DOS
0107 030000    JB 0000       ; Retour ar DOS

```

7.5.16 FONCTION DE OPEN FILE

La fonction OPEN permet d'ouvrir un fichier existant déjà sur le disque. Pour ouvrir un fichier voyez la fonction de MVE. L'ouverture d'un fichier en CP/M ou en MSX-DOS se fait par référence au FCB (File Control Block) du fichier. Remarque: le FCB dans le chapitre 5. Le FCB est une zone de 32 caractères implantée en mémoire et elle est utilisée par le programmeur. On peut ouvrir des fichiers soit en mode séquentiel qu'en mode direct. Avant d'ouvrir un fichier, le programmeur doit remplir les données de base du FCB qui représentent les données suivantes du FCB:

```

DR      Numéro de disque avec ou sans bit B
RWAVE   Mode du fichier en D octets.
FRT     Extension du fichier en 3 octets.
CI      Numéro de l'extension en mode séquentiel. En
        général, on y met toujours 0005, mais on peut
        ouvrir un fichier directement en un seul don't
        (1 Canal = 128 records = 1024)
RSCBIZ  Doit toujours être égal à 00.

```

Il est possible d'ouvrir un fichier existant (voir fonction CREATE) en positionnant le bit B dans l'octet DR du FCB. Le reste du FCB sera automatiquement rempli d'0 pour la fonction OPEN. Il s'agit des données suivantes:

```

DLBIZ  Indiquera la longueur du fichier en nombre de
        bytes.
DAID   Indiquera la date de création ou de dernière
        modification du fichier.
LIMS   Indiquera l'adresse de création ou de dernière
        modification du fichier sur les MS12.
DEVIO   Donnera l'identificateur du type de périphérique.
DIRLOC   Donnera la position du fichier dans le Directory.
STRLOC   Donnera le Start Cluster du fichier.
CURLCB   Donnera le Current Cluster.
CLOC     Donnera le décalage en plusieurs par rapport au
        début du fichier.

```

Les données DR indiquent record et RM indiquent numéro sont indiqués dans les tables 2.2.2. non-traduites par la fonction OPEN.

En entrée : La registre DE doit contenir l'adresse du FCB.
En sortie : La registre A et L contiendront 00 si le fichier a été ouvert. Si le fichier n'est pas ouvert sans erreur. Les registres A et L contiendront FF et le CARRY flag sera à 1 si l'ouverture n'est pas réussie par une erreur telle que:

- disque inexistant
- disque plein
- fichier inexistant

LE LIVRE DU DISQUE MSX - 163

Remarque: l'initialisation du FCB en 1600
Toutes les positions non-utilisées ont été
initialisées à 0 par les instructions DEFW.
Pour essayer ce programme, s'ouvrir une de
crée un petit fichier TEXT.ASC

7.5.18 FONCTION 11 SEARCH FIRST

La fonction 11 permet de rechercher dans le directory un
fichier dont le FCB est protégé par la registre DE, 51 et
le fichier existe dans le Directory, les 32 octets du directory
relatifs à ce fichier sont copiés dans le DMA buffer (défaut
= 0000) et les registres A et L sont positionnés à 00 et le
carry flag est mis à 0.

Si le fichier n'existe pas, les registres A et L retournent
le valeur FFH et le carry flag est mis à 1.

On peut utiliser les caractères de substitution avec le nom
du fichier présent dans le FCB. Dans ce cas, cette fonction
recherche le premier fichier, et puis le début du
Directory, coïncidant avec le nom présent dans le FCB.

En entrée : Le registre DE doit pointer sur le FCB du
fichier. Le FCB doit être composé avec l'appel
de cette fonction et peut contenir des
caractères de substitution.

En sortie : Les registres A et L contiennent 00 et le
fichier est trouvé dans le Directory. Les
registres A et L contiennent FF et le fichier
n'est pas trouvé dans le Directory. Le DMA
buffer contient les 32 octets du
Directory relative à ce fichier avec en plus
les indications suivantes:

DATA12 = Numéro d'entrée (n) que précède dans
le position FCBI12

DATA13 = Numéro du fichier (n) précédant dans
le position FCBI12

DATA14 = 0

DATA15 = Numéro de registre (12) ayant
présenté dans l'entrée DATA12. Cette
valeur est mise à 0 dans le
registre E.

Préserve : Aucun registre n'est préservé. Par contre,
l'adresse du FCB est sauvegardée dans la position
FCBDM du Ram pour être utilisée par la
fonction suivante SEARCH NEXT entrée 12.

Compatible : Oui

Exemple : Voir fonction suivante.

7.5.19 FONCTION 12 SEARCH FOR NEXT

La fonction 12 est complémentaire à la fonction 11
précédente. Elle permet de rechercher le prochain occurrence
dans le Directory du fichier dont le FCB est présent. Elle
est la fonction SEARCH FIRST précédente. Comme l'on peut

utiliser les caractères de substitution, les fonctions 11 et
12 forment une paire permettant entre autre de visualiser le
contenu du Directory. La fonction 11 recherche la première
occurrence du nom de fichier et des fonctions 12 successives
recherchant les occurrences suivantes.

En entrée : Aucun paramètre n'est nécessaire, mais cette
fonction doit être précédée de la fonction
11 qui fixe le FCB du/des fichiers recherchés.

En sortie : Voyez la fonction 11.

Préserve : Rien.

Exemple : Le petit programme va vous montrer comment
utiliser la fonction 11 ou 12 en langage
machine. La fonction 11 au début du programme
permet de fixer l'adresse du FCB et sera vu
plus tard. Tous les fichiers du Directory
sont énumérés y compris les fichiers
cachés.

C000 001A	DMA:	LD C,0A	place le leçon DMA
C002 110000		LD DE,0000	en 0000
C005 C07DF3		CALL 0F37D	
C008 0011	PRINT:	LD C,11	search first file
C00A 113C00		LD DE,FCB	
C00B C07DF3		CALL F37D	
C010 380F		JE C,END	pas trouvé -> END
C012 C02AC0		CALL PRINT	affiche son fichier
C015 0012	WAIT:	LD E,12	search next file
C01F C07DF3		CALL F37D	
C01A 380F		JE C,END	pas trouvé -> END
C01C C02AC0		CALL PRINT	affiche son fichier
C01F 18F4		JP NEXT	passé la recherche
C021 0000	END:	LD C,0	fin du programme
C023 C07DF3		JP F37D	
C02E 210180	PRINT:	LD 0,0001	affiche son fichier
C029 0A0C		LD 0,0C	affiche 12 caract.
C02B 3C	PRINT:	LD E,11H	affiche son fichier
C02C 23		INC R	affiche son fichier
C02D C5		KUSH 3C	affiche son fichier
C02E F3		PUSH H	affiche son fichier
C02F 0E02		LD C,2	affiche son fichier
C031 C07DF3		CALL 0F37D	
C034 01		POP H	
C035 01		POP 3C	
C036 10F3		CALL PRINT	
C038 000F	CRIF:	LD C,0F	affiche son fichier
C03A 1140C0		LD DE,0F	
C03D C07DF3		JP F37D	
C040 000A24	CR:	DEFB 00,0A,24	affiche son fichier
C045 00	FCB:	DEFB 00	affiche son fichier
C048 3F3F3F3F		DEFB 3F,3F,3F,3F	affiche son fichier
C04B 3F3F3F3F		DEFB 3F,3F,3F,3F	affiche son fichier
C04C 3F3F3F		DEFB 3F,3F,3F	affiche son fichier
C04F 00000000		DEFB 0,0,0,0	affiche son fichier
C053 00000000		DEFB 0,0,0,0	affiche son fichier
C057 00000000		DEFB 0,0,0,0	affiche son fichier
C05B 00000000		DEFB 0,0,0,0	affiche son fichier
C05F 00000000		DEFB 0,0,0,0	affiche son fichier
C063 00000000		DEFB 0,0,0,0	affiche son fichier

7.3.20 FONCTION 13 DELETE FILE

La fonction 13 permet d'effacer un fichier du Directory de la disquette. Cette opération n'est pas la même que celle du fichier mais suppose tout d'abord de connaître le numéro du fichier dans le Directory de la disquette. Elle est effectuée dans la FDI dans les cases prévues à cet effet. On peut effacer plusieurs fichiers d'un lot en utilisant les caractères de substitution. Le fichier à effacer sera effacé par son DCB dont l'adresse doit être fournie dans la registre DL.

- En sortie :** Le registre DE doit contenir l'adresse du FCB ou fichier à effacer.
- En sortie :** Les registres A et L contiendront DON si l'opération se déroule sans erreur. Les registres A et L contiendront FFH si l'opération se termine par une erreur telle que : la disquette n'est pas présente, le fichier n'existe pas.
- Préserve :** Aucun registre n'est préservé.
- Compatible :** Oui.
- Exemple :** Ce petit programme efface tous les fichiers portant l'extension .TMP

```

0100 0E13      ORG 0100
0102 210E01    START: L0 C13      | donellon Deloitte
0105 0E0E00    L0 DE,FCB        | DE = ELO
0108 0E0E00    CALL 0005        | Appel fonction
010B 0A        JP 0000          | Retour au Dos
010E 0A        DCR 00          | Décrémenter
0110 3F3F3F3F  DCFB '????????' | Nom du fichier
0113 3F3F3F3F  DCFB '????????' | Nom du fichier
0116 0E0E00    DCR 01          | L1 = TMP
0119 0E0E00    DCFB 0,0,0,0     | Bitacres FCB
011C 0E0E00    DCFB 0,0,0,0     |
011F 0E0E00    DCFB 0,0,0,0     |
0122 0E0E00    DCFB 0,0,0,0     |

```

7.3.31 FONCTION 14 SEQUENTIAL READ

La fonction 14 permet de lire un enregistrement de 128 octets étendu CP/M à partir d'un fichier et de transférer cet enregistrement en mémoire dans la OIA. Le fichier à lire doit avoir été préalablement ouvert par la fonction OPEN 10F. Le registre DE doit préciser l'adresse de ce FCB.

Les enregistrements sont lus séquentiellement à partir de l'enregistrement 0. Donc à chaque appel de cette fonction l'enregistrement suivant sera transféré vers la DTA. Les caractères obtenus par la lecture sont transférés automatiquement dans les positions LA (Curien) et EX (Extent) du FCB.

Par exemple, si l'on a une disquette contenant un enregistrement de 128 octets, on peut lire les données de ce

code de lecture et l'ouvrir avec le séquenceur de lecture.

L'enregistrement est lu séquentiellement par les zones EX et CR du DCB. EX doit être dans quel état ? L'Extent = 128 = 128 enregistrements de 128 octets se trouve l'enregistrement et CR doit être le numéro de l'enregistrement dans le lot. Extent = 0 à 127.

- En sortie :** Le registre DE doit pointer sur le fichier ouvert dans quel état ? L'Extent = 128.
- En sortie :** Les registres A et L contiendront DON si l'opération se déroule sans erreur. Les registres A et L contiendront OIH si l'opération se termine par une erreur telle que : le fichier n'existe pas, la disquette n'est pas présente, le fichier n'est pas ouvert.
- Préserve :** Cette fonction ne préserve aucun registre.
- Compatible :** Oui, d'ailleurs cette fonction est très proche de la fonction 14 de la disquette.

Exemple : Ce petit programme va vous montrer comment lire un fichier ASCII de votre disquette. Appelons ce fichier ASCII. Le programme lira le fichier ASCII et le transfèrera dans la DTA. Le fichier ASCII sera lu séquentiellement à partir de l'enregistrement 0. Le fichier ASCII sera lu séquentiellement à partir de l'enregistrement 0. Le fichier ASCII sera lu séquentiellement à partir de l'enregistrement 0.

```

0100 210E01    START: L0 DE,FCB | Copie séquentielle dans
0103 0E0E00    L0 DE,FCB        | la DTA
0106 0E0E00    L0 DE,FCB        |
0109 0E0E00    L0 DE,FCB        |
010B 0E0E00    L0 DE,FCB        |
010E 0E0E00    L0 DE,FCB        |
0110 0E0E00    L0 DE,FCB        |
0113 0E0E00    L0 DE,FCB        |
0116 0E0E00    L0 DE,FCB        |
0119 0E0E00    L0 DE,FCB        |
011C 0E0E00    L0 DE,FCB        |
011F 0E0E00    L0 DE,FCB        |
0122 0E0E00    L0 DE,FCB        |
0125 0E0E00    L0 DE,FCB        |
0128 0E0E00    L0 DE,FCB        |
012B 0E0E00    L0 DE,FCB        |
012E 0E0E00    L0 DE,FCB        |
0131 0E0E00    L0 DE,FCB        |
0134 0E0E00    L0 DE,FCB        |
0137 0E0E00    L0 DE,FCB        |
013A 0E0E00    L0 DE,FCB        |
013D 0E0E00    L0 DE,FCB        |
0140 0E0E00    L0 DE,FCB        |
0143 0E0E00    L0 DE,FCB        |
0146 0E0E00    L0 DE,FCB        |
0149 0E0E00    L0 DE,FCB        |
014C 0E0E00    L0 DE,FCB        |
014F 0E0E00    L0 DE,FCB        |
0152 0E0E00    L0 DE,FCB        |
0155 0E0E00    L0 DE,FCB        |
0158 0E0E00    L0 DE,FCB        |
015B 0E0E00    L0 DE,FCB        |
015E 0E0E00    L0 DE,FCB        |
0161 0E0E00    L0 DE,FCB        |
0164 0E0E00    L0 DE,FCB        |
0167 0E0E00    L0 DE,FCB        |
016A 0E0E00    L0 DE,FCB        |
016D 0E0E00    L0 DE,FCB        |
0170 0E0E00    L0 DE,FCB        |
0173 0E0E00    L0 DE,FCB        |
0176 0E0E00    L0 DE,FCB        |
0179 0E0E00    L0 DE,FCB        |
017C 0E0E00    L0 DE,FCB        |
017F 0E0E00    L0 DE,FCB        |
0182 0E0E00    L0 DE,FCB        |
0185 0E0E00    L0 DE,FCB        |
0188 0E0E00    L0 DE,FCB        |
018B 0E0E00    L0 DE,FCB        |
018E 0E0E00    L0 DE,FCB        |
0191 0E0E00    L0 DE,FCB        |
0194 0E0E00    L0 DE,FCB        |
0197 0E0E00    L0 DE,FCB        |
019A 0E0E00    L0 DE,FCB        |
019D 0E0E00    L0 DE,FCB        |
01A0 0E0E00    L0 DE,FCB        |
01A3 0E0E00    L0 DE,FCB        |
01A6 0E0E00    L0 DE,FCB        |
01A9 0E0E00    L0 DE,FCB        |
01AC 0E0E00    L0 DE,FCB        |
01AF 0E0E00    L0 DE,FCB        |
01B2 0E0E00    L0 DE,FCB        |
01B5 0E0E00    L0 DE,FCB        |
01B8 0E0E00    L0 DE,FCB        |
01BB 0E0E00    L0 DE,FCB        |
01BE 0E0E00    L0 DE,FCB        |
01C1 0E0E00    L0 DE,FCB        |
01C4 0E0E00    L0 DE,FCB        |
01C7 0E0E00    L0 DE,FCB        |
01CA 0E0E00    L0 DE,FCB        |
01CD 0E0E00    L0 DE,FCB        |
01D0 0E0E00    L0 DE,FCB        |
01D3 0E0E00    L0 DE,FCB        |
01D6 0E0E00    L0 DE,FCB        |
01D9 0E0E00    L0 DE,FCB        |
01DC 0E0E00    L0 DE,FCB        |
01DF 0E0E00    L0 DE,FCB        |
01E2 0E0E00    L0 DE,FCB        |
01E5 0E0E00    L0 DE,FCB        |
01E8 0E0E00    L0 DE,FCB        |
01EB 0E0E00    L0 DE,FCB        |
01EE 0E0E00    L0 DE,FCB        |
01F1 0E0E00    L0 DE,FCB        |
01F4 0E0E00    L0 DE,FCB        |
01F7 0E0E00    L0 DE,FCB        |
01FA 0E0E00    L0 DE,FCB        |
01FD 0E0E00    L0 DE,FCB        |
0200 0E0E00    L0 DE,FCB        |

```

```

D131 POPB      D1NZ PRINL      | Plifie la boucle
D133 10E1      JR  REAR        | Retour à lecture
D135 0E1D      CLDSE:  LD  C,1D  | dévacture fichier
D137 110091    LD  DE,FCB      |
D139 CD0500    CALL 0005      |
D13B E30000    JP  0000        | Retour au DOS.
D140 00000000 #CB:  DEFB 0,0,0,0 | Le FCB sera rempli
D144 00000000  DEFB 0,0,0,0 | par la premier
D148 00000000  DEFB 0,0,0,0 | argument
D14C 00000000  DEFB 0,0,0,0 |
D150 00000000  DEFB 0,0,0,0 |
D154 00000000  DEFB 0,0,0,0 |
D158 00000000  DEFB 0,0,0,0 |
D15C 00000000  DEFB 0,0,0,0 |
D160 00000000  DEFB 0,0,0,0 |
D164 00000000  DEFB 0,0,0,0 |

```

7.5.22 FONCTION 15 DEBUNTAL WRITE

Cette fonction transfère le contenu de la BIE vers un enregistrement de fichier dont le DCB est précisé par la registre DE. Le fichier doit avoir été préalablement ouvert par la fonction OPEN IDFI ou créé par la fonction CREATE IDFI. Les enregistrements seront écrits séquentiellement car le fichier n'est pas préalablement créé. Les octets CR (carriage) seront et de l'écriture du FCB.

Il est possible d'insérer linéairement un enregistrement X en indiquant son adresse la valeur des octets CR et IX avant l'appel de cette fonction. Rappelons qu'il y a toujours 128 octets dans un enregistrement CR et qu'il y a 128 enregistrements dans un EXTENT.

En entrée : La registre DE doit pointer sur le FCB du fichier. Le DTA doit contenir les 128 octets de l'enregistrement. Les octets CR et IX du FCB doivent être positionnés sur la valeur d'enregistrement que vous voulez écrire et l'écriture doit se faire sans écho.

En sortie : Les registres A et L contiendront 00H et l'opération s'est déroulée sans erreur. Les registres A et L contiendront D1H et l'opération s'est déroulée par une erreur telle que : disque plein - fichier non ouvert - etc. La registre IX contiendra l'adresse de FCB. La registre L contiendra l'adresse du DCB.

Preserve : Cette routine ne conserve aucun registre.

Compatible : Oui, d'ailleurs cette fonction n'est pas placée en MSX-DOS pour assurer la compatibilité avec la CP/M. En effet, la fonction MSX-DOS Random Block Write fonction 261 offre beaucoup plus d'avantages et s'exécute bien plus rapidement que celle-ci.

Exemple : Voici l'exemple de la fonction CREATE suivante.

7.5.23 FONCTION 16 CREATE FILE

La fonction CREATE FILE permet de créer un fichier qui n'existe pas encore dans le Directory du disque. Elle crée

avec un fichier dont la longueur est de 1 octet. Elle crée un fichier de 1 octet et l'écrit dans le fichier. Elle crée un fichier de 1 octet et l'écrit dans le fichier. Elle crée un fichier de 1 octet et l'écrit dans le fichier.

Si le fichier existe déjà et que la position FCB+2 est placée à 00H, il est supprimé; on perd ainsi toutes les informations qu'il contenait.

Si le fichier existe déjà et que la position FCB+2 est différente de 00H, le contenu du fichier est conservé et le FCB est complété avec les informations du Directory. Donc, la date et l'heure de création, la longueur du fichier et son cluster de départ sont conservés. Cependant, si vous voulez ajouter le fichier, il est bien sûr de positionner les octets de CR et de la valeur souhaitée, car cette opération ne laisse dans l'état initial.

Cette fonction transfère avec une seule fois les données de fichier qu'il s'agit de type séquentiel ou direct. Les paramètres de cette fonction et de la fonction d'enregistrement ICR et CR ou RCR sont placés dans le FCB après l'exécution de cette fonction.

Si le bit 0 du numéro de disque dans le FCB (DR) est à 1, la fonction va créer un fichier normal. L'octet Attribut du Directory D1H+11 sera égal à 00H et les bits du fichier se créent sans erreur lors des recherches dans le Directory (voir Directory dans le chapitre 4).

En entrée : La registre DE doit pointer l'adresse de FCB du fichier. Le DCB doit avoir été complété avant l'appel de cette fonction.

En sortie : Les registres A et L contiendront 00H et l'opération s'est déroulée sans erreur. Les registres A et L contiendront D1H et l'opération s'est déroulée par une erreur telle que : disque plein - fichier non ouvert - etc. La registre L contiendra l'adresse du DCB.

Preserve : Aucun registre n'est préservé.

Compatible : Oui, surtout que la technique des fichiers est la même que celle de la CP/M.

Exemple : Voici un programme de copie du fichier placé dans le répertoire courant vers un fichier dont le nom figure en deuxième argument. Si vous avez ce programme sous le nom COPY.

ADCOPY FILETIRE.ASC ARCHIVE.ASC

copiera le fichier FILETIRE.ASC présent sur le disque D1 vers un fichier ARCHIVE.ASC qui sera créé sur le disque A1.

0100 215000	START:	LD	DE, FCB1	Copie argument 1 vers FCB1
0103 114E01		FD	FCB1, 10	
0106 011000		LD	DE, FCB2	Copie argument 2 vers FCB2
0109 FDB0		LD	DE, FCB2	
010B 216C00		LD	DE, FCB2	
010E 117301		LD	DE, FCB2	
0111 011000		LD	DE, FCB2	
0114 ED80		LD	DE, FCB2	
0116 0E0F	OPEN:	LD	F, 0	Open fichier 1
0118 114E01		FD	DE, FCB1	
011B 000500		JP	0005	Si erreur, > 0000
011E 0A0000		JP	0000	
0121 0E18	CREATE:	LD	F, 18	Crée fichier 2
0123 117301		LD	DE, FCB2	
0126 C00500		CALL	0005	
0129 07		OR	A	Si erreur, > 0000
012A C20000		JP	0000	
012D 0E18	READ:	LD	F, 18	Lire un record du fichier 1
012F 118F01		LD	DE, FCB1	
0132 C00500		CALL	0005	
0135 07		OR	A	Si erreur, > 0000
0138 2008		JP	0008	
0139 0E18	WRITE:	LD	F, 18	Ecrit record dans fichier 2
013A 117301		LD	DE, FCB2	
013C F00B00		CALL	0005	
0140 07		OR	A	Si erreur, > 0000
0141 200A		JP	000A	
0143 0E10	CLOSE:	LD	F, 10	Fermeture Fichier 2
0145 117301		LD	DE, FCB2	
0148 C00500		CALL	0005	
014B C30000		JP	0000	Retour au DOS
014E 00	FCB1:	DEFB	00	FFB copie du 1er argument
014F 00000000		DEFB	0, 0, 0, 0	
0153 00000000		DEFB	0, 0, 0, 0	
0157 00000000		DEFB	0, 0, 0, 0	
015B 00000000		DEFB	0, 0, 0, 0	
015F 00000000		DEFB	0, 0, 0, 0	
0163 00000000		DEFB	0, 0, 0, 0	
0167 00000000		DEFB	0, 0, 0, 0	
016B 00000000		DEFB	0, 0, 0, 0	
016F 00000000		DEFB	0, 0, 0, 0	
0173 00	FCB2:	DEFB	00	FCB copie du 2ème argument
0174 00000000		DEFB	0, 0, 0, 0	
0178 00000000		DEFB	0, 0, 0, 0	
017C 00000000		DEFB	0, 0, 0, 0	

7.5.20 FONCTION 17 RENVERNE FILE

Fa fonction 17 permet de changer le nom d'un fichier spécifié par le FCB pointé par le registre DE dans le nom donné par ce même FCB rris en position FCB+16. La structure de ce FCB spécial devient donc:

Longi	1	2	3	4
0001	DR-FILENAME-EXT-00-00-00-00-DR-FILENAME-EXT-00-00-00-00			
7 type	ancien nom du fichier	dont bits 0-00H	Nouveau nom du fichier	dont bits 0-00H

L'octet DR 001001 doit obligatoirement être identique dans les deux portions du FCB. FILENAME, EXT peut contenir des caractères de substitution tant que l'ancien nom que dans le nouveau.

Dans l'ancien cas, les caractères de substitution permettent de trouver pour les fichiers du Directory respectant avec le nom l'ancien, tandis que dans le nouveau cas, les caractères de substitution indiquent qu'il faut utiliser les caractères de l'ancien nom à chaque position où on rencontre un '?'.
En entrée : Les registres DE doit pointer sur le FCB dont les 16 premières positions contiennent l'ancien nom et les suivantes le nouveau nom du fichier.
En sortie : Les registres A et E contiendront DON et le Carry flag sera à 0 si l'opération se déroule sans erreur, les registres A et E contiendront FCB en cas d'erreur. Si l'erreur est que le fichier à renommer n'est pas trouvé, alors le Carry flag sera également à 1.

Exemple :
En entrée : Le registre DE doit pointer sur le FCB dont les 16 premières positions contiennent l'ancien nom et les suivantes le nouveau nom du fichier.
En sortie : Les registres A et E contiendront DON et le Carry flag sera à 0 si l'opération se déroule sans erreur, les registres A et E contiendront FCB en cas d'erreur. Si l'erreur est que le fichier à renommer n'est pas trouvé, alors le Carry flag sera également à 1.
Preserve : Aucun registre n'est préservé.
Compliable : Oui
Exemple : Ce programme fait l'égalisation de la commande DABIC NAME 'JEU7.DAB' AS '7777.OLD'

0100 0E17	START:	LD	C, 17	Fonction 17
0102 11....		LD	DE, FCB00B	DE = FCB
0104 C00500		CALL	0005	Appel DOS
0106 07		OR	A	Si prob erreur
0109 C00000		JP	0000	Retour au DOS
010C 0E0F		LD	C, 0F	Si erreur, alléger
010E 113701		LD	DE, 0000	Message d'erreur
0110 F00500		CALL	0005	et
0112 C30000		JP	0000	Retour au DOS
0114 00	FCB00B:	DEFB	00	Disque par défaut
0116 4A15553F		DEFB	'JEU7'	Ancien nom
0118 20202020		DEFB	0, 0, 0, 0	
011A 224153		DEFB	'0000'	
011C 00000000		DEFB	0, 0, 0, 0	
011E 00	FCB00B:	DEFB	00	Disque par défaut
0120 2F2F2F2F		DEFB	'7777'	Nouveau nom
0122 20202020		DEFB	0, 0, 0, 0	
0124 0F4C3F		DEFB	'0000'	
0126 00000000		DEFB	0, 0, 0, 0	
0128 0000	ERRMSG:	DEFB	00, 00	GR LF
012A 45227273		DEFB	'Erreur'	Message d'erreur
012C 7573		DEFB	00, 0A, 0A, 20	FF LF FF
012E 000A0A24		DEFB	00, 0A, 0A, 20	FF LF FF

7.5.25 FONCTION 18 GET LOGGIN VECTOR

Cette fonction permet de connaître les disques présents sur votre système. Elle retourne dans le registre R1 le nom du disque D1 à l'adresse 1 indiquée la présence d'un disque en l'err. Fr bit 1 de R1 est réservé au disque D1, le bit 2 du disque D1 et ainsi de suite jusqu'au bit 1 pour le disque D1.

En entrée : Rien
 En sortie : La registre H est à 00H et la registre L contient la table des disques Dn-128.
 Préserve : Les registres C, D et E sont préservés.
 Compatible : Oui, mais en MSX-DOS, tous les disques présents sur le système sont toujours On-Line tandis qu'en CP/M, il peuvent être Off-Line.
 Exemple : Ce programme affiche les noms des disques présents sur votre système.

```

0100 0E18      ORG 0100
0102 000500   START: LD C,H
0105 0408     LD D,00H
0107 70       LD A,00H
0108 05       START: PUSH BC
0109 F5       PUSH AF
010A EAD1     AND 01
010C 2800     JR 2,START2
010E 0E09     PRINT: LD C,H
0110 112201   LD DE,MSX
0112 000500   CALL 0005
0114 217901   START2: LD HL,DISC
0116 34       INC HL
011A F1       POP AF
011B 0F       ARCA
011C 01       POP BC
011E 10E9     OPNZ START1
011F C30000   JP 0000
0121 4447371 MES: DEFB 'Disque '
0123 75A520   DEFB 0D,0A,20
0125 41       DBC: DEB 01
0127 3A207073 DEB 'i présent'
0129 9273454E
0132 74       DEFB 0D,0A,20
0133 000A24   DEFB 0D,0A,20

```

7.3.2a FONCTION 10 SET DEFAULT DRIVE NAME

Cette fonction permet d'obtenir, dans la registre A, le numéro du disque par défaut. C'est donc la fonction inverse de Select Disk IOCTL qui permet de définir le disque par défaut. La valeur 0 est attribuée au disque A1, 1 au disque B1 et ainsi de suite jusqu'à 7 pour le disque H1.

En entrée : Rien
 En sortie : Les registres A et L indiquent le numéro du disque par défaut.
 Préserve : Les registres C, D et E sont préservés.
 Compatible : Oui
 Exemple : Ce programme affiche le nom du disque par défaut.

```

0100 0E19      ORG 0100
0102 000500   START: LD C,H
0105 0441     LD D,00H
0107 321E01   LD A,0E9
0108 0E09     LD D,0E9
010A 11501    LD DE,MSX
010C 000500   CALL 0005
0112 C30000   JP 0000
0115 44452044 MES: DEFB 'Le disque '
0119 69737173
011B 45
011E 003A20   DBC: DEFB 0D,0A,20
011F 45737420 DEFB 'est le '
0123 444520
0125 44697371 DEFB 'disque '
0127 75A520
0129 20417220 DEFB 'par défaut '
0131 44452044
0133 7574
0137 000A0A24 DEFB 0D,0A,0A,20

```

7.5.27 FONCTION 1A SET DMA BUFFER ADDRESS

La fonction 1A permet de fixer l'adresse mémoire où sera situé le tampon de l'enregistrement appelé DMA BUFFER en CP/M ou DMA BUFFER en MSX-DOS. Rappelons que ce tampon est situé par défaut à l'adresse 0000H tant en MSX-DOS qu'en BASIC-DOS et que la fonction DISK RESET réinitialise ce tampon à cette adresse. Si vous déplacez le tampon DMA à l'adresse 0000H par exemple, toutes les fonctions liées ou agissant sur le disque se serviront de cette adresse comme tampon. Elle est particulièrement nécessaire en BASIC-DOS puisque la valeur par défaut 1000H ne contient pas du RAM dans l'environnement BASIC-DOS.

En entrée : La registre DE doit contenir l'adresse où l'on désire implanter le tampon DMA.

En sortie : Rien
 Préserve : Les registres C, D et E sont préservés.
 Compatible : Oui
 Exemple : Voir l'exemple de la fonction 11 (SELECT) ci-dessus.

7.5.28 FONCTION 1B SET ALLOCATION

La fonction 1B permet d'obtenir une série de paramètres concernant le disque dont le numéro est précisé dans la registre E ou 0 vaut pour le disque par défaut, 1 pour le disque A1, 2 pour le disque B1, etc.

En entrée : La registre E doit contenir le numéro du disque dont on veut obtenir les paramètres.
 En sortie : A donne le nombre de secteurs par cluster, BC donne la taille du secteur en nombre d'octets, DE donne le nombre de clusters réservés à l'utilisateur.

HL donne la nombre de aluatera qui restent libres.
 LY donne l'adresse du DCB de ce disque.
 LY donne l'adrasaa du premier octet de la FAT réservée à ce disque.
 B: le numero da disqua en entréa est invalide, la registre A contiendra FFH et le Carry flag sera à 1.

Préserva : Rien
 Compatible : NON. Cette fonction fournit les mêmes types de renseignements en CP/M mais par la biais d'une adresse donnée dans HL. Etant donné la différence de structure du système d'exploitation disqua, il était impossible de simuler la fonction CP/M en MSX-DOS. Bien que rarement employée en CP/M, la non-compatibilité de cette fonction peut être source de gros problèmes.

Exemple : Voici un programme donnant les paramètres du disqua ahoal an hexadecimal.

```

0100 DE09      START: DRG 0100
0102 118A01    LD C,09
0104 C05000    LD DE,QUEST 1 'Quel disque?'
0106 0E01      INPUT: LD C,01 1 'Obtient la
0108 C05000    CALL 0005 1 réponse
010B 5F        GETALL: LD E,A 1 Appel fonction
010E 0E1B      GETALL: LD C,1B 1 Get Allocation
0110 C05000    CALL 0005
0113 3C        CHECK: INC A 1 Test si erreur
0114 200A      JR NZ,START1 1 non -> START1
0116 0E09      ERROR: LD C,9 1 Affiche message
0118 119B01    LD DE,ERR 1 d'erreur,
011B C05000    CALL 0005 1
011E 1B00      STAR1: JR START 1
0120 C631      ADD A,31 1 Affichage des
0122 FDE5      PUSH 1V 1 différents
0124 DDE5      PUSH 1X 1 paramètres
0126 E5        PUSH HL 1 précédés d'un
0127 D5        PUSH DE 1 message.
012B C5        PUSH BC 1
0129 32BF01    LD (SEC1),A 1
012C 11AB01    LD DE,NSC 1
012F C06701    CALL PRINT 1
0132 11C101    LD DE,SECS 1
0135 C06701    CALL PRINT 1
0138 C1        POP BC 1
0139 C06C01    CALL HEXA 1
013C 11D601    LD DE,NCLUS 1
013F C06701    CALL PRINT 1
0142 C1        POP BC 1
0143 C06C01    CALL HEXA 1
0146 11ED01    LD DE,NFCLU 1
0149 C06701    CALL PRINT 1
014C C1        POP BC 1
014D C06C01    CALL HEXA 1
0150 110002    LD DE,DPB 1
0153 C06701    CALL PRINT 1
0156 C1        POP BC 1
0157 C06C01    CALL HEXA 1
015A 111502    LD DE,FAT 1

```

```

015B C06701    CALL DEB701
0160 C1        POP BC 1
0161 C06C01    CALL HEXA 1
0164 C30000    JP 0000 1 Retour au DOS
0167 0E09      PRINT: LD C,9 1 Sous routine
0169 C30500    LD JP 0005 1 d'ajustation
016C 7B        LD A,B 1 Sous routine
016D C07101    CALL HEX1 1 d'affichage an
0170 79        LD A,C 1 hexadecimal
0171 F5        PUSH AF 1 du registre BC
0172 0F        RRCA 1
0173 0F        RRCA 1
0174 0F        RRCA 1
0175 0F        RRCA 1
0176 C07A01    CALL HEX2 1
0179 F1        POP AF 1
017A E60F      HEX2: AND OF 1
017C C630      AND A,30 1
017E FE3A      CP 3A 1
0180 3B02      JR C,HEX3 1
0182 C607      AND A,7 1
0184 5F        LD E,A 1
0185 0E02      LD C,02 1
0187 C30500    JP 0005 1
018A 0D0A      QUEST: DEFB 0D,0A
018C 5075656C  DEFB 'Quel disque ? '
0190 20646973
0194 71756520
0198 3F202E    ERR: DEFB 0D,0A
019B 0D0A      DEFB 'Numéro erroné'
019D 4E736D82  DEFB 0D,0A
01A1 726F2065  DEFB 'Saas./aluater 1 '
01A5 72726F6E
01A9 B224      NSEC: DEFB 0D,0A
01AB 0D0A      DEFB 'Saas./cluater 1 '
01AD 5345637E
01B1 2E2F636C
01B5 75737465
01B9 72202020
01BD 2A20      SEC: DEFB 0D,24
01BF 0024      SECS: DEFB 0D,0A,'Taille saasur : a'
01C1 0B0A5461
01C5 696C6C6A
01C9 20207365
01CD 63746973
01D1 72203A20
01D5 24      NCLUS: DEFB 0D,0A,'Cluater/diaque : '
01D6 9D0A436C
01DA 75737465
01DE 72732F64
01E2 69737175
01E6 60202A20
01EA 24      NFCLU: DEFB 0D,0A,'Cluater libre : '
01EB 000A436C
01EF 75737465
01F3 7273206C
01F7 69627265
01FB 73202A20
01FF 24      DPB: DEFB 0D,0A,'Adrasaa DPB 1 '
0200 000A4164
0204 72657373
0208 65204450

```


020C 12202020
 0210 20202020
 0214 24
 0215 00004167 FAT: DEFB 00,0A,1'Adresse FAT 1 2'
 0219 72657375
 021D 45204631
 0221 5T202020
 0225 20202020
 0229 24

7.5.29 FONCTION 1C NON UTILISEE EN MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s'agit de la fonction GET WRITE PROTECT VECTOR.

7.5.30 FONCTION 1D NON UTILISEE EN MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s'agit de la fonction GET WRITE PROTECT VECTOR.

7.5.31 FONCTION 1E NON UTILISEE EN MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s'agit de la fonction GET FILE ATTRIBUTES.

7.5.32 FONCTION 1F NON UTILISEE EN MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s'agit de la fonction GET DISK PARAMETER ADDRESSES.

7.5.33 FONCTION 20 NON UTILISEE EN MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s'agit de la fonction GET/SET USER CODE.

7.5.34 FONCTION 21 RANDOM READ

Cette fonction permet la lecture d'un enregistrement d'un fichier à accès direct dont le contenu sera transféré vers le tampon DTA. Le fichier devra avoir été préalablement ouvert par la fonction OPEN (00F). Le fichier est sélectionné par son FCB dont l'adresse doit être placée dans le registre DE. L'enregistrement qui sera lu sera déterminé par les octets RN (FCB133 à FCB136). Ces octets ne seront pas testés par cette fonction et doivent donc toujours être

placés par le programmeur avant chaque appel de cette fonction. Le saut ou enregistrement est toujours une longueur fixe de 128 octets en CP/M. Cette fonction a été implémentée en MSX-DOS uniquement pour le rendre compatible avec le CP/M mais l'usage de la fonction MSX-DOS RANDOM BLOCK READ 121 est beaucoup plus rapide et avantageux.

En entrée : Le registre DE doit pointer sur le FCB du fichier.
 Les octets RN du FCB déterminent le numéro de l'enregistrement qui sera lu.

En sortie : Les registres A et L contiennent la valeur 00 si l'opération s'est déroulée sans erreur.
 Les registres A et L contiennent la valeur 01 si l'opération s'est terminée par une erreur.
 Il y a une lecture du data de fin de fichier.
 Le registre H contient l'adresse du FCB.
 Les registres D et E contiennent l'adresse de DCS.
 Aucun registre n'est préservé.

Compatible : Oui, mais utilisation de la fonction 27 RANDOM BLOCK READ.
 Exemple : Voir fonction suivante.

7.5.35 FONCTION 22 RANDOM WRITE

Cette fonction permet l'écriture d'un enregistrement dans un fichier à accès direct. Le fichier est sélectionné par son FCB dont l'adresse doit être placée dans le registre DE. Le fichier devra avoir été ouvert par la fonction OPEN (00F) et le tampon DTA devra être rempli avec les données de l'enregistrement à écrire. L'appel de cette fonction, l'enregistrement qui sera écrit sera déterminé par les octets RN (FCB133 à FCB136). Ces octets ne seront pas testés par cette fonction et doivent donc toujours être placés par le programmeur avant chaque appel de cette fonction. Le saut ou enregistrement est toujours une longueur fixe de 128 octets en CP/M. Cette fonction a été implémentée en MSX-DOS uniquement pour le rendre compatible avec le CP/M mais l'usage de la fonction MSX-DOS RANDOM BLOCK WRITE 121 est beaucoup plus rapide et avantageux.

En entrée : Le registre DE doit pointer sur le FCB du fichier.
 Le tampon DTA doit contenir les données de l'enregistrement.

En sortie : Les octets RN du FCB déterminent le numéro de l'enregistrement qui sera écrit.
 Les registres A et L contiennent la valeur 00 si l'opération s'est déroulée sans erreur.
 Les registres A et L contiennent la valeur 01 si l'opération s'est terminée par une erreur.

Préserve : Aucun registre n'est préservé.
 Compatible : Oui, mais utilisation de la fonction 26 RANDOM BLOCK WRITE.

Exemple : Ce programme écrit les enregistrements 0, 3 et 9 du fichier TEST1.TST pour les sauvegarder dans le fichier TEST2.TST.

C000	001A	SETPM1:	ORG	C000	
C002	1100B0		LD	C,1A	
C005	C00500		LD	DE,C0B000	Ajuste l'adresse DTA
C008	0E0F	OP4N11	CALL	0005	en D000 pour
C00A	1171C0		LD	C,0F	travail en bas/c.
C00D	C00500		LD	DE,FC01	Duivre 14511.151
C010	B7		CALL	0005	
C011	C25B00		OR	A	el erreur -> END
C014	0E04	OPEN2:	JP	N1,END	
C019	1194C0		LD	C,0F	Duivre TEST2.T51
C019	C00500		LD	DE,4CB2	
C01C	B7		CALL	0005	
C01D	C25B00		OR	A	el erreur -> END
C020	2163C0		JP	N1,END	
C023	00	REC1:	LD	H,L,TAB.4	transfère adresse
C024	23		LD	C,0111	de record vers
C024	46		INC	H,L	FCB1+21 à partir
C026	23		LD	B,01H1	d'une table de
C027	03		INC	H,L	records à copier
C028	70		LD	A,0	
C029	A1		DR	C	
C02A	C054C0		JP	Z,CLOSE	
C02B	00		DEC	BC	
C02C	04392C0		LD	14CB1+211,BC	
C032	68		PUSH	H,L	
C033	0E21	READ:	LD	C,21	Lit la record
C038	1171C0		LD	DE,FC01	
C038	C00500		CALL	C005	
C03B	B7		OR	A	el erreur -> CLOSE
C03C	C25B00		JP	N1,CLOSE	
C03F	01	REC2:	POP	H,L	transfère record
C040	4C		LD	C,H,L1	de destination de
C041	23		INC	H,L	la table vers
C042	46		LD	B,H,L1	4CB2+21
C043	23		INC	H,L	
C044	0D43B7C0		LD	1FCB2+211,BC	
C046	46		PUSH	H,L	
C049	0422	WRITE:	LD	C,22	Ecrit record lu
C04B	0194C0		LD	DE,4CB2	dans le fichier 2
C04B	C00500		CALL	D000	
C051	C1		POP	H,L	
C052	B7		OR	A	el gae d'erreur -> REC1
C053	C023C0		JP	Z,REC1	
C054	0E10	CLOSE:	LD	C,10	trans fichier 2
C05B	1194C0		LD	DE,FCB2	
C05B	C00500		CALL	0005	
C05F	0E00	AND:	LD	C,0	Retour au Base
C060	C00500		J4	0005	
C063	08000700	TABLE:	DE4W	4,7	Table des records
C067	03000100		0446	3,1	à copier.
C06B	05000700		04FW	5,7	
C06B	4444		DEFW	044FF	FF44=fin de table
C071	00	FCB1:	DB	0	Nue. disque
C072	54453344		DB	'1BDT1 T51'	Non fichier
C074	31209020				
C07A	545354				
C07D	00000000		DB	0,0,0,0	
C081	00000000		DB	0,0,0,0	
C085	00000000		DB	0,0,0,0	
C084	00000000		DB	0,0,0,0	
C085	00000000		DB	0,0,0,0	

C091	00000000		DB	0,0,0,0	
C095	00		DB	0	
C096	00	4CB2:	DB	0	thue. disque
C097	54455354		DB	'TEST2 T51'	Non fichier
C099	32202020				
C09F	545354				
C0A2	00000000		DB	0,0,0,0	
C0A6	00000000		DB	0,0,0,0	
C0AA	00000000		DB	0,0,0,0	
C0AE	00000000		DB	0,0,0,0	
C0B2	00000000		DB	0,0,0,0	
C0B6	00000000		DB	0,0,0,0	
C0B7	00		DB	0	

7.3.36 FONCTION 23 GET FILE SIZE

La fonction 23 permet de connaître le nombre d'enregistrement de 128 octets présents dans un fichier dont la FCB est donné par le registre DE. Le résultat est placé dans les octets RH du 4CB e.4.0. en position 4CB+33, FCB+34 et FCB+35. Cette fonction est écrit de la table du fichier en mémoire d'octets pour obtenir le nombre de records. Cette fonction est particulièrement utile pour ajouter des enregistrements à la suite d'un ancien enregistrement d'un fichier à accès direct séquentiel. Il suffit en effet d'ouvrir d'abord le fichier, puis d'utiliser cette fonction 23 pour insérer le nombre du prochain enregistrement à la suite. Attention qu'il s'agit en fait d'un nombre virtuel d'enregistrement: il est en effet possible de créer un fichier "fictif" avec plusieurs "enregistrements" ayant respectivement les numéros 4 et 123 et des "trous" entre-temps. Dans ce cas, la fonction retournera dans les octets RH le valeur 124 comme s'il y avait déjà 124 enregistrements réellement écrits dans le fichier alors qu'il n'y en a que deux.

En entrée : le registre DE doit contenir l'adresse du 4CB du fichier. Ce fichier peut être ouvert ou fermé. Le FCB doit être initialisé avant l'appel de cette fonction.

En sortie : Les positions 4CB+33, 4CB+34 et FCB+35 donneront le nombre de records qui ont le dernier record du fichier. Si ce nombre est inférieur à 255, il sera écrit. Si le nombre est supérieur à 255, le registre DE donnera l'adresse du FCB. Les registres A et L seront placés à 00 si l'opération se déroule sans erreur ou à FF si l'opération se termine par une erreur (le fichier n'existe pas).

Préserve : Aucun registre n'est préservé.
 Compatible : Oui.
 Exemple : Ce programme lit le nombre de records de 128 octets du fichier placé avec le nom argument dans le programme. Si vous saisissez le programme sous le nom REC23.COM, passer simplement ceci :

A>REC23 0:4FCHIER.T51

pour connaître la taille de fichier FICHER,101
présent sur le disque B1 en mode de record
appelé en hexadécimal.

```

ORG 0100
0100 215000 START: LD 18,3C      ; Copie l'adresse
0103 118A01 LD DE,FC9      ; dans le FCB
0106 011000 LD BC,10      ;
0109 ENB0 LDIR          ;
010B 0E23 SIZE: LD 4,23      ; Appel fonction 23
010D 115A01 LD DE,FC9      ;
0110 CD0500 CALL 0005      ;
0113 87 OR A          ; Si erreur,
0114 C20000 JP NZ,0000      ; -> DOS
0117 3A7001 PRINT: LD A,18H?  ; A = FN2
011A CD3401 CALL HEXA      ; affiche A en hexa
011D 3A7C01 LD A,18H1      ; f = AN1
0120 CD3401 CALL HEXA      ; affiche A en hexa
0123 3A7901 LD A,18H01      ; f = AN0
0126 4D3401 CALL HEXA      ; affiche f en hexa
0129 0E49 ENDI: LD C,09      ; Affiche message
012B 114D01 LD DE,CRLF      ; cf
012E 4D0500 CALL 0005      ; retour au DOS
0131 330000 Jd 0008      ;
0134 F3 HEXA: PUSH AF      ;
0135 0F RRcA          ;
0136 0F RRCA          ;
0137 0F RRCA          ;
0138 0F FFA          ;
0139 4D3Dd1 CALL HEX1      ;
013C F1 POP AF          ;
013D 660F HEX1: AND 0F      ;
013F C630 ADD A,30      ;
0141 F3A CP 3A          ;
0143 3B02 JP C,HEX2      ;
0145 4407 ADD A,7        ;
0147 8F HEX2: LD E,A        ;
014B 0E02 LD C,D2        ;
014E 330500 JF 0008      ;
014D 4D207265 CRLF: DEFB 'M recorda' ; Message final
0151 3A6F7264 LD 0,0A,0A,24 ;
0153 73 DEFB 00          ;
0156 0D0A0A24 FCB: DEFB 0D,0A,0A,24 ; FCB dont les 16
015A 00 DEFB 0          ; premiers octets
015d 00000000 DEFB 0,0,0,0 ; sont copies de
015e 00000000 DEFB 0,0,0,0 ; premier enregistrement
0163 00000000 DEFB 0,0,0,0 ;
0167 00000000 DEFB 0,0,0,0 ;
0168 00000000 DEFB 0,0,0,0 ;
016F 00000000 DEFB 0,0,0,0 ;
0173 00000000 DEFB 0,0,0,0 ;
0176 00000000 DEFB 0,0,0,0 ;
0177 00000000 DEFB 0,0,0,0 ;
0178 00 AN0: DEFB 0        ; Octets RN du FCB
017C 00 AN1: DFB 0         ;
017D 00 AN2: DFB 0         ;
017E 00 AN3: DFB 0         ;

```

7.3.37 FONCTION 24 SET RANDOM RECORD

La fonction 24 est une fonction de CP/M qui est officiellement supportée par le MSX-DOS. Cependant, dans une dilution version de configuration disque que j'ai de penser une erreur ou bug s'est glissée dans la portion de code destinée à cette fonction par MICROSOFT. Il est évident qu'elle ne peut pas fonctionner. Cependant, nous allons quand même expliquer son fonctionnement pour ceux qui auront une version de FDISK avec elle. Nous fournissons une petite routine qui réalise la même fonction pour les effets.

Le but de cette fonction est de traduire le numéro de record assigné en un numéro de record à accès direct. En effet, lorsqu'on manipule un fichier séquentiel, le programme ne garde pas toujours la liste de numéro de record où l'on est arrivé. Ainsi, cette fonction traduit les octets S2-CR-IFCB+14, FCB+12, FCB+32 déterminant le numéro de record en mode séquentiel en une équivalence dans les octets RN-IFCB+35, FCB+34 et FCB+33 utilisés en mode direct.

En entrée : Le registre DE doit pointer sur le FCB du fichier ouvert qui a déjà été manipulé par des fonctions SEQUENTIAL READ 1141 et SEQUENTIAL WRITE 1151.

En sortie : Dans le registre RN-IFCB+33, FCB+34 et FCB+35 devraient contenir le numéro de record. Le registre IX contient l'adresse du FCB.

Précondition : Rien.

Compatibilité : Oui, elle devrait être compatible avec le bug de la ROM. Actuellement les octets FCB+33 et FCB+31 retournent respectivement 01-00 si l'octet FCB+35 est nul (la valeur normale).

Exemple : Nous illustrons ici une routine qui sert à traduire la place de cette fonction si c'est le cas. Cette routine est entièrement pédagogique. Elle signifie que vous pouvez l'implémenter si vous le souhaitez.

```

XXXX 00 RNDREC: 0000 XXXX
FDE1 PUSH DE      ; Sauve le FCB dans
F4420 POP IX      ; registre IX
F4420 LD C,(IX+20) ; C = FCB+32 CR
F4420 LD 0,(IX+0C) ; 0 = FCB+12 EN
F4420 LD 4,(IX+0E) ; 4 = FCB+14 F2
F4420 LD 0,0       ; D = 0
F4420 CBL 0        ; C = C + 2
F4420 SRL E        ; E
F4420 RRC B        ; B
F4420 RRC C        ; C
F4420 LD 11Y+21,C  ; FCB+33 = C
F4420 LD 11Y+21,B  ; FCB+34 = B
F4420 LD 11Y+23,E  ; FCB+35 = E
F4420 RET          ;

```

7.3.38 FONCTION 25 NON UTILISÉE EN MSX-DOS

Retourne 00 dans les registres A et L si pas de...

registres D, 0 et E. En CP/M, il s'agit de la fonction RESET DISK DRIVE.

7.5.39 FONCTION 26 RANDOM BLOCK WRITE (MSX-DOS)

Voici une fonction exclusivement MSX-DOS qui remplace à elle seule les fonctions CP/M SEQUENTIAL WRITE (15) et RANDOM WRITE (22). Elle est beaucoup plus rapide et autorise de fixer la longueur et le nombre des enregistrements qui vont être écrits à votre guise.

Cette fonction écrit dans le fichier spécifié par le FCB pointé par le registre DE un nombre de records spécifié par le registre HL dont les données se trouvent dans le tampon DTA.

Le numéro du premier record à écrire doit être placé dans les octets RN du FCB (FCB+33, FCB+34, FCB+35 et FCB+36). Ces octets RN seront automatiquement incrémentés du nombre de records écrits si l'opération se déroule sans erreur. Cela permet de quitter l'opération avec le FCB prêt pour une écriture suivante (mode séquentiel). Si vous désirez travailler en mode direct, il faudra placer le numéro du record à écrire dans les octets RN avant chaque appel de cette fonction.

La taille des records à écrire n'est plus fixée à 128 octets comme en CP/M mais est programmable. Il suffit pour cela de placer la taille désirée (de 1 à 65535 octets) dans les positions RECSIZ du FCB (FCB+14 et FCB+15).

En entrée : le registre DE doit indiquer l'adresse du FCB.
Le registre HL doit indiquer le nombre de records à écrire.
Les octets RECSIZ du FCB doivent indiquer la taille du record.
Les octets RN du FCB doivent indiquer le numéro du record de départ.
Le tampon DTA doit contenir les données du/des records à écrire.

En sortie : le registre A indiquera 00 si l'opération se déroule sans erreur ou 01 dans le cas contraire. Les octets RN du FCB seront incrémentés du nombre de records écrits.
Les registres HL et BC indiqueront le nombre de records écrits.
Le registre IX donnera l'adresse du DCS et le registre IY l'adresse du FCB.

Préserve : Aucun registre n'est préservé.
Compatible : NON. Cette fonction n'existe pas en CP/M.
Exemple : Ce programme permet de sauvegarder sur un fichier le contenu complet de la mémoire réservée au DOS.

0100 0E1A	GETDTA:	LD	C,10	1	Tampon DTA = 0000
0102 110000		LD	DE,0000	1	
0105 C00500		CALL	0005	1	
0108 0E16	CREATE:	LD	C,16	1	Création fichier
010A 113701		LD	DE,FCB	1	MEMORY,DMP
010D C00500		CALL	0005	1	
0110 B7		OR	A	1	SI erreur,
0111 C20000		JP	HZ,0000	1	-> DOS
0114 210000	WRITE:	LD	HL,0	1	Numéro record = 0
0117 225001		LD	IFCB+211,HL	1	
011A 225A01		LD	IFCB+231,HL	1	longu. record = 1
011D 23		INC	HL	1	
011E 22A501		LD	IFCB+0E1,HL	1	Nbr. record = TDP
0121 2A0A00		LD	HL,1000A1	1	
0124 113701		LD	DE,FCB	1	Ecriture TOP rec.
0127 0E26		LD	C,26	1	
0129 C00500		CALL	0005	1	Fermeture fichier
012C 0E10	CLOSE:	LD	C,10	1	
012E 113701		LD	DE,FCB	1	
0131 C00500		CALL	0005	1	
0134 C30000		JP	0000	1	Retour au DOS
0137 00	FCB:	DB	0	1	FCB
0138 1D45A04F		DB	'MEMORY'	1	
013C 3259					
013E 444050		DB	'DMP'	1	
0151 00000000		DB	0,0,0,0	1	
0155 00000000		DB	0,0,0,0	1	
0159 00000000		DB	0,0,0,0	1	
015D 00000000		DB	0,0,0,0	1	
0161 00000000		DB	0,0,0,0	1	
0165 00000000		DB	0,0,0,0	1	
0169 00		DB	0	1	

Voici encore une fonction exécutée vers MX1-D05 qui remplace P elle seule les fonctions CP/M SEQUENTIAL READ 1141 et RANDOM READ 1211. Plus est beaucoup plus rapide et autonome de 1144 la longueur et le nombre des enregistrements qui vont lire les à votre guise.

Cette fonction 111 dans le fichier spécifié par la PCB pointée par le registre DE un nombre de records spécifiés par le registre IL et transfère les données dans le temps OTA.

Le numéro du premier record à lire doit être placé dans les octets RN du PCB (PCB+33, PCB+34, PCB+35 et PCB+36). Ces octets RN seront automatiquement incrémentés du nombre de records lus si l'opération se déroule sans erreur. Cela permet de quitter l'opération avec le PCB prêt pour une lecture suivante (mode séquentiel). Si vous devez travailler en mode direct, il faudra placer le numéro du record à lire dans les octets RN avant chaque appel de cette fonction.

La lettre des records P lire s'est plus élevée à 128 octets dans le CP/M soit est programmable. Il suffit pour cela de placer la lettre Paires (de 1 à 127) dans les octets des positions RECORD du PCB (PCB+4 et PCB+13).

En entrée : Le registre DE doit indiquer l'adresse du PCB. Le registre IL doit indiquer le nombre de records P lire. Les octets PCB+12 du PCB doivent indiquer la taille du record. Les octets RN du PCB doivent indiquer le numéro du record de départ.

En sortie : Le registre IL indiquera 00 si l'opération se déroule sans erreur ou 01 dans le cas contraire. Les octets RN du PCB seront incrémentés du nombre de records lus. Les registres IL et BC indiqueront le nombre de records effectivement lus. Le registre IL contiendra l'adresse du PCB si le registre IL l'adresse du PCB.

Préserve : Aucune registre n'est préservée.
Compatible : NON. Cette fonction n'existe pas en CP/M.
Exemple : Vous savez que tout programme MX1-D05 en charge en passant simplement le non de programme après l'indicateur du DOS (>). Par exemple, un programme s'installant toujours à l'adresse 0100H du début de la IFA. Cet exemple va vous montrer comment charger un programme à une adresse différente. Nous expliquerons le programme LOAD.COM et pour l'utiliser nous passerons après l'indicateur du DOS ceci :

ANLOAD NON-OU-PROGRAMME 235F

Le premier argument sera le nom du programme ou de fichier à charger et le second argument l'adresse mémoire où vous voulez l'installer. La taille du fichier sera extraite du PCB pour connaître la taille du record à charger.

0100 215000	START:	LD	HL, 005C	Copy de premier
0103 115001		LD	DE, PCB	argument dans le
0106 011000		LD	BC, 10	FCB.
0109 C000		LD		
010B 215701	LOAD:	LD	HL, RESULT+1	Convertit second
010E 116000		LD	DE, 0040	argument en
0111 C04001		CALL	BIN	à lire dans
0114 2P		DEC	IL	RESULT.
0115 C04001		CALL	BIN	
0118 C05001	SETDT:	LD	DE, RESULT	Place le temps
011C 0E1A		LD	C, 1A	OTA à l'adresse
011E C00500		LD	HL, 0005	RESULT.
0121 115A01	OPEN:	LD	DE, FCB	Ouvre le fichier
0124 0E0F		LD	C, 0F	
0126 C00500		CALL	0005	
0129 07		DR	A	Et c'est tout.
012A C20000		JP	NZ, 0000	-> 005
012D 2A4A01	LOAD:	LD	HL, (PCB+10)	Taille fichier ?
0130 226A01		LD	(PCB+0E), HL	Taille record.
0133 210000		LD	HL, 0	Metre 0 dans
0136 221B01		LD	(PCB+21), HL	numéro de record
0139 227D01		LD	(PCB+23), HL	octets RN.
013C 23		INC	HL	HL = 100, record
013D 115A01		LD	DE, PCB	à lire
0140 0E27		LD	C, 27	Appel fonction
0142 C00500		CALL	0005	
0145 C30000		JP	0000	à l'adresse du DOS.
0148 C04B01	BIN:	CALL	BIN1	Routine get
014B 1A	BIN1:	LD	A, 10E1	convertit 10E1 et
014C D630		SUB	30	à 10E1 ce octet
014E 443A		CP	3A	à l'adresse
0150 3602		JR	C, 30H2	
0152 D601		SUB	7	
0155 13	BIN2:	INC	DE	
0158 E0E1		RLD		
015B C0		RLD		
015E 0000	RESULT:	DEFP	0	Adresse chargement
015A 00	PCB:	DEFP	0	PCB dont les 16
015B 00000000		DEFP	0,0,0,0	premier octet
015F 00000000		DEFP	0,0,0,0	second octet
0163 00000000		DEFP	0,0,0,0	troisième octet
0167 00000000		DEFP	0,0,0,0	quatrième octet
016B 00000000		DEFP	0,0,0,0	cinquième octet
016F 00000000		DEFP	0,0,0,0	sixième octet
0173 00000000		DEFP	0,0,0,0	septième octet
0177 00000000		DEFP	0,0,0,0	huitième octet
017B 00000000		DEFP	0,0,0,0	neuvième octet

7.3.41 FONCTION 26 RANDOM WRITE WITH ZERO FILL

Cette fonction est similaire à la fonction RANDOM WRITE 1221 à l'exception que, lors d'une extension du fichier, tous les records non encore écrits sont remplis de zéros.

L'exemple de cette fonction à la place de la fonction 22 fait que le fichier ne contiendra aucun "trou" c'est à dire des records non-écrits qui pourraient tromper un programme de lecture et ses records non-écrits disparaîtront à tout.

Pour que ce fichier soit rempli sans trou, il faut absolument utiliser la fonction 26 pour chaque écriture dans le fichier.

Exemple : Si vous créez un fichier et que vous le remplissez avec cette fonction en écrivant les records dans l'ordre suivant 1, 2, 3, 4, 21, alors les records de zéro 0, 3, 4, 5, 6, 7, 8, et 10 à 20 contiendront 128 zéros chacun.

Note d'exemple : Voir fonction 22.
Compatible : Oui.

7.3.42 FONCTION 29 NO FUNCTION (MSX-DOS)

Cette fonction est réservée pour de futures extensions du MSX-DOS. Elle retourne la valeur 0 dans le registre A et préserve les registres C, D et E.

7.3.43 FONCTION 2A GET DATE AND TIME (MSX-DOS)

Cette fonction retourne la date et l'heure et l'allumage de votre système pour les MSX's sans horloge interne (MSX1). Pour les MSX's 2, elle retourne la date et le temps (heure et minute seulement) stockés dans le registre électronique protégé par batterie libre de leur propriétaire.

En entrée : Rien.
En sortie : IL retourne l'année sous la forme YYYY en décimale. Ex 1987 = 07C3, H = 07, t = C3
D retourne la mois en décimale H-121.
B retourne l'heure en décimale 10-23 MSX21
C retourne les minutes en décimale 10-59 MSX21
E retourne le jour du mois en décimale H-311.
A retourne le jour de la semaine : 1 pour dimanche, 2 pour lundi, 3 pour mardi, 4 pour mercredi, 5 pour jeudi, 6 pour vendredi, 7 pour samedi.

Préserve : Aucune registre n'est préservée.
Compatible : Non.
Exemple : Ce programme va vous donner le jour de la semaine en français et la date dans l'ordre correct, que votre système soit ASCII ou EBCDIC.

0100 DE2A	START:	LD	E,2A	Appel fonction
0102 D0500		CALL	0005	
0103 E5		PUSH	HL	Sauve l'année
0104 D5		PUSH	DE	Sauve Mois et Jour
0107 217F01	NOM2R:	LD	H,1JOUR	Multiplie code de
010A B7		ADD	P,A	Jour de la semaine
010B B7		ADD	A,A	pour B:
010C B7		ADD	A,A	Modulo 7 résolu
010D B5		LD	L,1	l'adresse de la
010E 6F		JR	NC,NOM31	table.
010F 3001		INC	H	
0111 24		LD	B,B	Double B fois pour
0112 060B	NOM31:	PUSH	BC	afficher le non de
0114 C5	NOM32:	LD	E,DL1	jour de la semaine
0115 5E		PUSH	HL	
0116 23		INC	HL	
0117 E5		PUSH	HL	
011B 0E02		LD	C,02	
011A ED0500		CALL	0000	
011D E1		POP	HL	
011E C1		POP	BC	
011F 10F3		CALL	NOM32	
0121 0E04	NOM33:	LD	C,04	Affiche virgule et
0123 117B01		LD	DE,NER1	un espace.
0124 D0500		CALL	0000	
0128 21	DAT31:	POP	DE	Rappel Mois / Jour
012A D5		PUSH	DE	Sauve Mois
012B 78		LD	B,A	Jour -> A
012C D0F401		CALL	PRDU	Affiche jour
012F F1	DAT32:	POP	AF	Rappel Mois -> A
0130 D05401		CALL	PRDU	Affiche Mois
0133 3E13	DAT41:	LD	A,13	Affiche 10
0135 ED0F01		CALL	DINDEC	
013B D0A001		CALL	PRINT	
013D E1		POP	HL	
013E 016C07		LD	BC,076C	Rappel Année
013F B7		DN	A	BC = 1900
0140 ED4F		BC	HL,BC	Clear carry
0142 70		LD	A,t	L = année - 1900
0143 D0F401		CALL	DINDEC	Affiche année
0144 D0E001		CALL	PRINT	
0145 0C05		LD	C,05	
0149 117B01		LD	DE,CRLE	Affiche CR-LF
014E D07000		CALL	000B	
0151 C30000		JP	0000	Retour au DOS
0154 D0F401	PRDU:	CALL	DINDEC	Affiche décimale
0157 D0E001		CALL	PRINT	et unité.
015A 1620		LD	E,1	Affiche "-"
015C C30500		JP	0005	et retour.
015F 1E30	BINDEC:	LD	E,30	Convertit décimale
0161 D0A001	PINDEC:	SUD	0A	en 2 chiffres
0163 3B03		JR	C,BINDEC	seulement.
0165 1E		INC	E	
0166 1BFF		JR	DINDEC	E = 10001 EN 30
0168 C004	BINDEC:	ADD	P,A	
016A C050		ADD	A,30	A = (ANNO) OR 30
016C E9		RET		
016D 15	PRINT:	PUSH	AF	Routine d'affichage
016E 0E02		LD	C,02	
0170 D05000		CALL	0005	Affiche E
0173 F1		POP	AF	

```

0174 SF      LO E,B      | After b B
0175 C30500   JB 0005    |
0176 2C2021 MESI DEFB ' , ' |
0177 00000A24 CRLF DEFB 00,0A,0A |
0178 8A990A01 JGHR DEFB 'Dianehe' |
0183 0E630A05
0187 4C15AFA1 DEFB 'Lundi',0,0,0
018B 69000900 DEFB 'Marti',0,0,0
018F 1d617261 DEFB 'Mercredi'
0193 69000900 DEFB 'Jeudi',0,0,0
0197 40657263 DEFB 'Vendredi'
019B 72155169 DEFB 'Samedi',0,0,0
019F 1A657364 DEFB 'Dimanche'
01A3 09000000
01B1 5ee5e64 DEFB 'Ventadi'
01B7 72155169 DEFB 'Samedi',0,0
01BF 531ed65
01B3 14690000

```

1.5.44 FUNCTION 28 SET, DATE 1985-08-11

Cette fonction localise la date donnée dans les registres 01100 et 01101 dans les positions 01100 et 01101. Elle est appelée par le programme principal en F21A, la routine calcule elle-même le numéro du jour de la semaine et l'heure en F24E et le nombre de jours écoulés depuis le 1/1/1980 et l'heure en F24C et F24D. Elle modifie également la date du nombre de jours qui suitant que l'heure est donnée ou son F22C = F24E. Il vous dispose d'un horaire électronique 1985/12/1, la date du mois servie dans la seconde de ce composant qui est protégé par batterie. Une autre, cette fonction va modifier la valeur de la date que vous posez dans les registres et également toute donnée en plaçant FF dans le registre B.

Le registre B doit contenir l'année. Les années entre 1980 et 2099 sont seules compatibles avec les données et se peuvent être écrites en entrée et sortie.

B doit contenir le mois.

C doit contenir le jour.

A indique 00 si la date est valide.

A indique FF si la date n'est pas valide.

Les positions 01100 et 01101 de la date seront écrites à jour de l'heure que les données indiquent et 01100.

Préserve : Aucune registre n'est préservé.

Compilation : Non.

Exemple : Voici comment installer la date du 30/9/87

```

0100 21C301 START LO HL,07C3 | 07C3 = 1987
0103 1609 LD 0,9 | 9 = MOIS
0106 181E LD E,1E | 1E = JOUR
0107 0E2B LD C,2B |
0109 C00500 CALL 0005 | Appel fonction
010C C30000 JB 0000 | Retour au DOS

```

1.5.45 FUNCTION 28 SET, DATE 1985-08-11

Cette fonction retourne l'heure qu'il est en heures, minutes et secondes dans les registres 01100 et 01101. Elle est appelée par le programme principal en F21A, la routine calcule elle-même le numéro du jour de la semaine et l'heure en F24E et le nombre de jours écoulés depuis le 1/1/1980 et l'heure en F24C et F24D. Elle modifie également la date du nombre de jours qui suitant que l'heure est donnée ou son F22C = F24E. Il vous dispose d'un horaire électronique 1985/12/1, la date du mois servie dans la seconde de ce composant qui est protégé par batterie. Une autre, cette fonction va modifier la valeur de la date que vous posez dans les registres et également toute donnée en plaçant FF dans le registre B.

En entrée : Rien

En sortie : H retourne les heures de 0 à 23.

M retourne les minutes de 0 à 59.

S retourne les secondes de 0 à 59.

D est prévu pour retourner les secondes de secondes lorsqu'une horloge n'est pas présente sur le jour en MSX.

Préserve : Aucun registre n'est préservé.

Compilation : Non.

Exemple : Ce petit programme affiche l'heure en permanence dans le coin de l'écran.

```

0100 114901 START ORG 0100
0103 0E09 LD DE,CLB | CLB = cursor off
0105 C00500 START: CALL 0005 | HOME
0108 0E7C TIME: LO C,2C | Appel fonction
010A C00500 CALL 0005 |
010B 05 PUSH OF | Save saveSet
010E 05 PUSH HL | Save stack
010F 7C HL: LD A,H | A = heures
0110 C03101 CALL BINDEC | A heures -> décimal
0113 5E4B LD B,H | A heures -> décimal
0115 C4301 CALL PRINT |
0118 F1 MINUT: POP HL | Rappel minutes
0119 7D LD B,L |
011A C03101 CALL BINDEC | Minutes -> décimal
011D 3E27 LD B,"" | Affiche
011F C4301 CALL PRINT |
0122 01 SECON: POP DE | Rappel secondes
0123 7A LD B,D |
0125 C03101 CALL BINDEC | Secondes -> décimal
0127 3E22 LD A,"" | Affiche
0129 C4301 CALL PRINT |
012C 1808 LD E,8B | Curseur Home
012E C30501 JP START | et occurrence
0131 1E30 LD E,30 | Convertit A décimal
0133 C40A BINDEC: SUB 0A | de S = secondes
0135 3B03 JR C,BINDEF2 | A = unités
0137 1C INC E |
0138 1B79 JR BINDEF1 |
013A C43A BINDEF2: ADD A,3A |
013C F5 PUSH RF |
013D 0E02 LD C,02 | Affiche dizaines
013F C00500 CALL 0005 |
0142 F1 POP AF | Affiche unités
0143 5F PRINT: LD B,B | Affiche A
0144 0E02 LD C,02 |
0146 C30501 JP 0005 |
0149 0C1B7B35 C: DEFB 0C,1B,"e3" | CLB = ESC-e-3
014D 24

```

7.5.46 FUNCTION 2F SET TIME (MSX-DOS)

Cette fonction permet d'ajuster l'horloge interne de votre MSX2 en plaçant l'heure donnée dans les registres décrits ci-dessous dans l'horloge électronique. Bien entendu un contrôle de validité des heures, minutes et secondes est effectué.

En entrée : H doit contenir les heures de 0 à 23,
 L doit contenir les minutes de 0 à 59,
 D doit contenir les secondes de 0 à 59,
 C pour contenir les centièmes de secondes
 lorsqu'une horloge électronique de cette
 précision n'est pas en MSX.
 En sortie : A contiendra 00 si l'heure est valide,
 A contiendra FF si l'heure n'est pas valide.
 Préserve : Aucun registre n'est préservé.
 Compatible : Non.
 Exemple : Voici comment placer 12h00'00" dans l'horloge
 électronique.

	ORG 0100	
0100 240C	START: LD H,0C	; H = 12 heures
0102 2E00	LD L,0	; L = 0 minute
0104 1A00	LD D,0	; D = 0 seconde
0106 0E20	LD C,20	; Appel fonction
0108 CD0500	CALL 0005	
010B C30000	JP 0005	; Retour au DOS

7.5.47 FUNCTION 2E REPLY/VERIFY FLAG (MSX-DOS)

Cette fonction permet d'établir ou de remettre à zéro le drapeau de vérification (Verify flag). Lorsqu'il est établi, ce drapeau provoque que tous les secteurs du disque seront relus de façon automatique par le MSX-DOS pour s'assurer que les données ont été écrites correctement sur le disque. Cette méthode de travail est beaucoup plus sûre puisque vous évitez ainsi le risque que les données écrites sur le disque sont relues mais s'écrit comme inconveniant de ralentir un peu l'opérateur d'écriture. Le drapeau sera de ce drapeau supprime cette lecture.

En entrée : E doit contenir 00 pour retirer le drapeau à zéro.
 E doit contenir autre chose que 00 pour établir le drapeau.
 En sortie : Le code placé dans le registre E sera transféré vers la position RAM F50D. Cette position peut être consultée pour connaître l'état du drapeau.
 Préserve : Les registres G, D et E sont préservés.
 Compatible : Non.

Cette fonction permet de lire le disque par secteur plutôt que par entrelacement. Il s'agit donc d'une lecture en mode "physique" et non pas "logique". La fonction en elle-même lit sur le disque donné par le registre L le nombre H de secteurs à partir du secteur de départ DE vers le tampon DTA.

En entrée : H doit contenir le nombre de secteurs que vous voulez lire (de 1 à 255).
 L doit contenir le numéro du disque sur lequel se trouvent les secteurs à lire (0=A1, 1=B1, 2=C1, 3=D1).
 DE doit contenir le numéro du premier secteur à lire. La répartition des secteurs sur le disque dépend du type de disque si peut être trouvé dans l'annexe A.

En sortie : C indiquera le nombre de secteurs lus.
 DE indiquera le premier secteur lu.
 L indiquera l'adresse du DTA.
 En cas d'erreur, vous recevrez le message d'erreur habituel du MSX-DOS ou du BASIC suivant l'arrangement sous lequel vous travaillez, par exemple:
 ON BASIC-DOS : Disk Error
 on MSX-DOS : Drive not ready error... Abort, Retry, Ignore
 Il n'y a pas d'indication dans un registre ou un flag de cette erreur.
 Préserve : Seul le registre DE est préservé s'il n'y a pas d'erreur.
 Compatible : Non.
 Exemple : Voici comment lire les 7 secteurs du Directory d'un disque de 360K en D200H de la mémoire. Voyez l'exemple à part entière les paramètres pour un autre type de disque.

	ORG 0100	
0100 0E1A	START: LD C,1A	; Place le tampon
0102 110002	LD DE,D200	; DTA en D200H.
0105 CD0500	CALL 0005	
0108 2E00	REPLY: LD L,0	; Disque A1
010A 2A07	LD H,7	; 7 secteurs à lire
010C 110500	LD DE,8	; premier secteur à lire
010F 0E2F	LD C,2F	; Appel fonction
0111 E00500	CALL 0005	
0114 C30000	JP 0000	; Retour au DOS

Ce programme ne fait que lire les 7 secteurs du Directory à la place de D200H à l'adresse de la mémoire. A vous de placer une routine de vérification en lieu et place.

Cette fonction permet d'écrire sur le disque en mode physique c.à.d en donnant le numéro de secteur. La fonction écrit H secteurs sur le disque logique L à partir du secteur DE. Le contenu des secteurs à écrire doit avoir été préalablement installé dans le tampon DTA.

En entrée : H doit contenir le nombre de secteurs que vous désirez écrire (de 1 à 255).

L doit contenir le numéro du disque sur lequel se trouvent les secteurs à écrire. 1 = 0xA, 1401, ... = 0x1

DE doit contenir le numéro du premier secteur à écrire. La numérotation des secteurs sur le disque commence à 0. Le numéro maximum dépend du type de disque et peut être trouvé dans l'annexe A.

En sortie : C indiquera le nombre de secteurs écrits. DE indiquera le premier secteur écrit. IX indiquera l'adresse du DCB.

En cas d'erreur, vous recevrez le message d'erreur habituel du MSX-DOS ou du BASIC suivant l'environnement sous lequel vous travaillez, par exemple :
on BASIC DOS : Disk offline
or MSX DOS : Drive not ready error...
Abort, Retry, Ignore

Il n'y a pas d'indication dans un registre ou un flag de cette erreur.

Préserve : Seul le registre DE est préservé s'il n'y a pas d'erreur.

Compatible : Non.

Exemple : Attention à cette fonction. Comme elle écrit sur le disque en physique, il est très facile d'écraser le contenu de celui-ci. Pour expérimentier cette fonction, utiliser un disque ne contenant aucun fichier utile ! Soyez particulièrement attentif à ne pas écraser le secteur 0, les tables FAT et le Directory (Voyez l'annexe A pour connaître le numéro de ces secteurs).

Ce petit programme va lire le secteur 0 du disque A1, vous demander votre nom et ré-écrire ce secteur en plaçant votre nom en position 3 à 9.

0100 0E1A	START:	LD	C, 1A	1	Place DTA en 200H
0102 210002		LD	DE, 200	1	
0105 CD0500		CALL	0005	1	
0108 2E00	READ:	LD	I, 0	1	lecture disque A1
010A 2601		LD	H, 1	1	1 secteur
010C 110000		LD	DE, 0	1	Numéro 0
010E 0E2F		LD	C, 2F	1	
0111 CD0500		CALL	0005	1	
0114 0E09	NON:	LD	C, 09	1	Affiche message
0117 114201		LD	DE, GUEST	1	demandant votre
011A CD0500		CALL	0005	1	nom
011D 0E0A	INPUT:	LD	C, 0A	1	lecture de votre
011F 115001		LD	DF, BUF	1	réponse en 8 car.
0122 3E08		LD	A, 8	1	maximum dans BUF
0124 12		LD	(DE), A	1	
0125 CD0500		CALL	0005	1	
0128 215201	COPY:	LD	HL, BUF+2	1	Copie votre réponse
012B 110302		LD	DE, 203	1	dans tampon DTA + 3
012E 010800		LD	BC, 08	1	
0131 ED80		LDIR		1	
0133 2E00	WRITE:	LD	I, 0	1	Ecrit sur disque A1
0135 2601		LD	II, 1	1	1 secteur
0137 110000		LD	DE, 0	1	Numéro 0
013A 0E3D		LD	C, 3D	1	
013C CD0500		CALL	0005	1	
013F C30000		JP	0000	1	Retour au DOS
0142 0C	GUEST:	DEFB	DC	1	C15
0143 564F5152		DEFB	'VDTRG NOM ? '	1	
0147 43204E4F					
014B 4D203F20					
014F 24					
0150 0000	BUF:	DEFB	0, 0	1	
0152		DEFB	8	1	

7.6 Direct Bios access

Le CP/M permet des accès directs à certaines routines du BIOS en ajoutant un décalage à l'adresse d'implantation du CP/M en mémoire qui est indiquée en position 1 et 2 de la mémoire.

Le MSX-DOS a réalisé aussi ce genre d'accès à certaines routines mais du au différence de manipulation des fichiers, une partie limitée seulement de ces routines peut être accédée en MSX-DOS.

Voici le tableau comparatif entre les accès directs du CP/M et du MSX-DOS.

OFFSET	MSX-DOS	CP/M	FONCTION
00H	oui	oui	Retour à l'indicatif du DOS
03H	oui	oui	Retour à l'indicatif du DOS
06H	oui	oui	Fonction console status (08)
09H	oui	oui	Fonction console input (01)
0CH	oui	oui	Fonction console output (02)
0FH	NON	oui	Fonction list output (05)
12H	NON	oui	Fonction Aux. output (04)
15H	NON	oui	Fonction Aux. input (03)
18H	NON	oui	Move to track 00
1BH	NON	oui	Select disk drive
1EH	NON	oui	Set track number
21H	NON	oui	Set sector number
24H	NON	oui	Set DMA address
27H	NON	oui	Read selected sector
2AH	NON	oui	Write selected sector
2DH	NON	oui	Return List status
30H	NON	oui	Sector translate subroutine

Exemple : Pour retourner à l'indicatif du DOS, il y a maintenant 3 possibilités :

- 1) JP 0000 C'est l'utilisation du point de sortie de la page 0.
- 2) LD C,00
CALL 0005 C'est l'utilisation de la fonction 0.
- 3) LD HL,(1)
JP (HL) C'est l'utilisation du Direct bios call

Exemple : Pour lire un code du clavier, il y a deux possibilités :

- 1) LD C,02
CALL 0005 C'est l'utilisation de la Fonction 2.
- 2) Utiliser le Direct BIOS call. Il faut, pour cela, mettre dans BC la valeur de l'offset du tableau ci-dessus, appeler une routine qui va charger dans HL l'adresse d'implantation du DOS (donnée par les position 1 et 2) et ajouter BC à HL pour ensuite sauter à (HL). Dans l'exemple ci-dessous, chaque caractère tapé au clavier sera affiché deux fois à l'écran.

```

0100 010300  RKB0:  ORG 0100
0103 0D0E01  LD BC,09      ; BC = Offset table
0106 010C00  WSCR:  CALL DBIOS  ; pour Console Input
0109 0D0E01  LD BC,0C      ; BC = Offset table
010C 1BF2    CALL DBIOS  ; pour Console Output
010E 2A0100  JR START    ; Recommence.
0111 09      LD HL,(1) ; HL = Adresse du DOS
0112 E9      ADD HL,BC  ; Ajoute Offset à HL
                JP (HL) ; Saut à (HL)

```

Vous aurez certainement remarqué qu'il est beaucoup plus simple d'employer les fonctions standard décrites plus avant dans ce chapitre.

L'éditeur MSX-DOS et les fichiers BATCH

8.1 L'éditeur du MSX-DOS

8.1.1 Généralités

Si vous avez déjà travaillé en MSX-Basic, vous aurez certainement apprécié le confort de son éditeur plein-écran. En effet, la correction d'une faute de frappe est aussi simple que possible, il suffit de déplacer le curseur jusqu'à l'endroit de l'erreur grâce aux touches de déplacement du curseur et puis de taper le caractère correct.

En MSX-DOS par contre, l'éditeur n'utilise pas la technique du plein-écran mais un éditeur-ligne compatible avec le MS-DOS qui est le système d'exploitation des PC compatibles.

Vous aurez déjà remarqué que pour valider une commande du DOS, il suffisait d'enfoncer la touche RETURN comme en BASIC. De même, pour corriger une commande tant que l'on n'a pas fait RETURN, la touche BS (Backspace) et la touche <- (Curseur à gauche) ont toutes deux pour effet d'effacer le caractère précédent le curseur et puis de positionner le curseur à l'emplacement effacé.

Par contre, les touches suivantes ne fonctionnent pas comme en BASIC:

ARTRY DWERTY

SLP (DEL) suppression caractère

INS code insertion

EFF/DEP (CLR/HOME) Effacement écran / Curseur en haut gauche

et les trois touches de déplacement du curseur vers le haut, le bas et à droite avec lesquelles vous avez sans doute déjà remarqué qu'il n'est pas possible de déplacer le curseur sur une ligne supérieure ou inférieure ou encore à droite.

L'éditeur-ligne du MS-DOS et du MSX-DOS a la particularité de ne traiter qu'une ligne à la fois et de disposer d'une mémoire de la dernière ligne posée. Cette mémoire porte le nom de "TEMPLATE" dans le jargon anglais du MS-DOS. Pour la facilité d'expression, nous emploierons le terme "Tampon d'entrée" dans la suite de ce chapitre.

Ce tampon d'entrée se remplit automatiquement lorsque vous posez un texte quelconque après l'indicatif A> du DOS. Ainsi, si vous posez le texte ci-dessous suivi de RETURN, nous allons pouvoir expérimenter quelques possibilités:

A>REM édité telle pour essayer l'éditeur
 A> L'écrit A> s'est allié sur la ligne suivante dès que vous avez enfoncé le code RETURN. Ensuite, alléguant le touche "Curseur vers le bas" et vous verrez que le contenu de "l'ampoule d'entrée" s.d. la ligne que vous avez émise précédemment sera réajustée allié et le curseur restera en fin de ligne pour vous permettre de le modifier ou de le valider par RETURN.

Essayez maintenant le touche "Curseur vers le haut" et vous voyez que la ligne qui vient d'être réajustée disparaît immédiatement à l'exception de A>.

Si vous enfoncez maintenant de façon répétitive la touche "Curseur à droite", vous voyez réapparaître la ligne caractéristique par caractère de même que la touche "Curseur à gauche" la fait disparaître caractère par caractère.

A>REM Petit lexé pour essayer l'éditeur
 A>REM édité tel

Enfin, "Curseur vers le bas" pour réajuster la ligne complète et la ligne RETURN.

La touche SELECT suivie d'un caractère fait apparaître le contenu de "l'ampoule d'entrée" jusqu'au caractère spécifié après SELECT mais le tampon d'entrée ne sera pas modifié.

A>REM Petit lexé pour essayer l'éditeur
 A> L'éditeur: SELECT à p
 A>REM Petit lexé

Enfonçant SELECT + e se la ligne précédente se modifiera en

A>REM Petit lexé pour essayer l'

Nous allons maintenant évaluer des fonctions qui jouent également sur le contenu de "l'ampoule d'entrée" sans avoir d'effet visible à l'écran.

D'abord la touche SUP qui agit dans le "tampon d'entrée" la caractéristique sur lequel le curseur se trouve. Pour visualiser le résultat appuyez sur "Curseur vers le bas".

A>REM Petit lexé pour essayer l'éditeur
 A>SUP + "Curseur vers le bas"
 A>REM Petit lexé pour essayer l'éditeur

Si vous enfoncez plusieurs fois de suite la touche SUP, vous constaterez successivement la suppression de caractères. Le tampon d'entrée n'est pas modifié.

La touche EFF agit dans le tampon d'entrée les caractères depuis l'endroit du curseur jusqu'au caractère spécifié après EFF mais non caractère précédent. Pour visualiser le résultat appuyez sur "Curseur vers le bas". Ici aussi, le tampon d'entrée ne sera pas modifié.

A>REM Petit lexé pour essayer l'éditeur
 A>EFF + p et puis "Curseur vers le bas"
 A> pour essayer l'éditeur

La touche DEF permet d'envoyer la ligne telle qu'elle est allié à l'écran dans le "tampon d'entrée". Un caractère à s'allier en fin de ligne et le curseur apparaît sous le premier caractère de la ligne sans indiquer A>. Cette touche DEF agit donc identiquement à la touche RETURN à l'exception que la rommée s'est pas émise mais simplement rangée dans le "tampon d'entrée". RETURN reçoit l'exécution la commande allié.

Enfin, il reste la touche INS qui agit à l'écran du lexé dans une ligne déjà posée.

A>REM Petit lexé pour essayer l'éditeur
 ISELECT + p

A>REM
 IINS + Volet se p
 ISUP = appuyez la "e" de Petit
 ICurseur vers le bas
 A>REM Volet se petit lexé pour essayer l'éditeur

TABLEAU RÉSUMÉ DES TOUCHES DE FONCTION

dans le lexé explicatif précédent, nous n'avons employé qu'une seule touche par caractère des fonctions analysées. Dans le tableau ci-dessous, nous allons indiquer les touches touchées ou combinées de touches qui provoquent le même résultat. Le code placé entre parenthèses est le libellé de cette touche ou les claviers DIRECTY.

TOUCHE	FONCTION
Curseur vers bas CTRL-↓	Affiche le contenu du tampon d'entrée
Curseur vers haut CTRL-↑ CTRL-↑ ESC	Efface de l'écran la ligne allié et modifier le tampon d'entrée.
Curseur à droite CTRL-→	Allie le prochain caractère du tampon d'entrée.
Curseur à gauche CTRL-← CTRL-← DEL	Efface le dernier caractère allié de l'écran sans modifier le tampon d'entrée.
DEF (HOME) CTRL-K	Rangé la ligne allié dans le tampon d'entrée.
SELECT car. CTRL-X car.	Allie le tampon d'entrée jusqu'au car. spécifié.
EFF (CLR) car. CTRL-L car.	Efface dans le tampon d'entrée car. spécifié.
SUP (DEL)	Supprime le caractère d'un caractère dans le tampon d'entrée.
INS CTRL-P	Insère le caractère de code insertion. Arrêt du code insertion par une autre commande

8.1.2 Lete caradellu su da conta fide

Il existe cinq écoles qui fonctionnent non seulement au cours de l'édition, mais aussi durant l'exécution. Il n'y a pas que les écoles de la région.

Le plus important de ces codes est certainement CTRL-C qui est l'équivalent de CTRL-SHOP de Basic. De plus, CTRL-C permet de lancer l'exécution d'un programme MS-DOS et le retour à l'invite A> du DOS.

[illegible]

```

Le code CILC-9 prévoyait une liste clientèle c l'écrit en c
muraux muraux ven l'implication. Si, en muraux, muraux
deCelle l'implication. Le Directeur c l'écrit en c, muraux, muraux
empêcher une des. Nous muraux c l'écrit en c, muraux, muraux
d'abord CILC-9 pour passer DILC-9 l'écrit en c, muraux, muraux
vers que dans le premier une la muraux DILC-9 c l'écrit en c, muraux, muraux
l'implication celle du Directeur l'écrit en c, muraux, muraux
sauf la l'écrit en c, muraux, muraux

```

Le code CIM-10 émis par l'Institute of Medicine (IOM) est le code CIM-10.

Finalemant, le code CTRL-J provoque un retour au début de la ligne si celle-ci n'est pas en cours de calcul. La commande postée, cela peut être utile pour couper ou glisser des lignes entre commandes dans l'éditeur. Mais attention, le code CTRL-J ne sert pas à insérer de saut de ligne et ne modifie pas le contenu d'une ligne.

8.2 Les fichiers de commandes ou batch files

Il existe deux types particuliers de lichens en fait : **BOUL** et

Le premier est celui dont l'attention se larde en ger, COM en
est composé d'un programme en langage machine qui s'installe
lorsque c'est l'adresse COM de la mémoire RAM, il peut être
exécuté simplement en utilisant le bouton de l'icône
à l'écran de COM avec l'adresse de COM de
la mémoire RAM.

Le second type de fichier porte l'extension .bat et ce
contient ce qui programme un langage machine sans l'usage
telle de noms de programmes ou de commandes du DOS. Il peut
être exécuté aussi en entrant directement le nom du fichier
à l'invite.

Le larin 841 provient du nom anglais MACH qui signifie 'LOT' et est reconnu car le M41-801 pour faciliter que soit reconnu une liste l'ail de programme plutôt qu'un seul programme de lecture.

Lorsque vous êtes le son d'un licier, Bah, le client
attende le Dilemme du dinosaure pour liquider ce cas cili cas
l'attention, COM est avec l'attention, NT. de ce côté, il
fait l'absence d'être de donner un sens non A un licier, COM
et, Bah car cas cili cili ne cili cili plus il dit l'absence
le cas cili cili.

Si le P&G veut faire passer la direction du disque, le client va analyser et l'assignation porte le nom „CUM de „BAI. Si l'assignation est „CUM, le client sera chargé en DIOH de la carte avec le nom appliqué par le client sera chargé de la carte avec un JUMP DIOH.

Si l'extension du préfixe est .DRI, alors la fonction concernée sera bien sûr différente. Le fichier en question sera le fichier `libdri.so` et concernera des choses différentes. Ici, le nom du programme est chargé en une commande du DDB via des instructions de la bibliothèque `libdri.so`.

[illegible]

Voici d'abord un exemple de recréation d'un petit fichier de commande que vous appellerez COME24.BAT. Le rôle de ce fichier de commande sera de vous donner le code, l'heure et votre nom sur un MSX2 et le contenu de la disquette du disque par défaut.

Pour cliquer ou cliquer, vous pouvez utiliser un éditeur
tel que nous nous en sommes peut-être, nous allons
utiliser le commande COPY CON CONTEU.DAT qui permet
d'envoyer vers le fichier CONTEU.DAT ce que nous taperons
au clavier. Après le faire, nous cliquons ligne, nous
enregistrons le fichier RETURN et nous nous enregistrons.

l'écriture des lignes, nous possédons CTRL-Z itin de fin de fichier sur la ligne vierge suivante suivi d'un mode RETURN. A ce moment, tout ce que nous aurons fait sera écrit sur le disque dans le fichier CONTENU.BAT

```
A>COPY CON CONTENU.BAT
DATE
TIME
DIR /M
~Z
A>
```

Realisez maintenant, le mode CTRL-Z itin de fin de fichier s'affiche par "1". C'est la même chose que le MSX-DOS lorsque par le CP/M ou le MS-DOS d'ailleurs pour représenter les modes CONTROL.

Deuxième constatation, si nous avons écrit une suite de caractères sur une ligne particulière, il est impossible de réécrire à cette ligne sur l'écran car nous employons COPY qui va écrire un tableau des caractères (voir chapitre 7.3). Il faudra attendre toute l'opération depuis le début.

Troisième constatation, nous pouvons maintenant afficher le contenu de notre fichier en passant comme suit :

```
A>TYPE CONTENU.BAT
DATE
TIME
DIR /M
A>
```

Et nous voyons que tout ce que nous avons fait s'affiche à l'exception du CTRL-Z qui sert de marque de fin de fichier. Notre fichier tient maintenant tout, nous allons pouvoir l'utiliser en passant tout simplement son nom sans l'extension .BAT après l'indicateur du DOS.

```
A>CONTENU          <- Vous voyez ceci et RETURN
A>DATE             <- Automatique
Current date is Sun 21-06-87
Enter new date:    <- Automatique
A>TIME            <- Si date OK press RETURN
A>TIME            <- Automatique
Current time is 17:38:21
Enter new time:    <- Automatique
A>DIR /M           <- Si heure OK press RETURN
A>DIR /M           <- Automatique
MSXDOS SYS COMMAND.COM CONTENU.BAT <- Automatique
3 Files 351232 bytes free <- Automatique
A>
```

Nous pouvons constater que chaque commande s'affiche d'abord automatiquement derrière l'indicateur du DOS et puis qu'elle s'exécute. Si nous sommes satisfaits de la date et de l'heure nous n'avons qu'à taper RETURN et le système poursuit l'exécution du fichier de commandes.

Voilà un autre exemple qui peut être utile pour ceux qui ont un MSX à deux lecteurs tels que les PHILIPS VG8255 ou VG9250. Il fonctionne également sur les machines à un seul disque, mais nécessite alors des changements de disquettes. Il sert à créer une disquette contenant le MSX-DOS à partir d'une disquette vierge. Voici d'abord comment le créer :

A>COPY CON DOSDISK.BAT

```
REM CREATION D'UNE DISQUETTE MSX-DOS
REM -----
```

REM ATTENTION REPONDEZ BIEN A LA QUESTION SUIVANTE

```
VERIFY ON
FORMAT
COPY A:MSXDOS.SYS B:
COPY A:COMMAND.COM B:
~Z
A>
```

Dans cet exemple vous pouvez constater l'utilité de la commande REM qui sert à afficher des remarques qui seront visibles lors de l'exécution du fichier. Remarque également qu'on peut introduire une ligne blanche pour avoir la présentation lors de l'exécution. Pour entrer les lignes s'il y a d'abord le disquette à formater dans le lecteur B il suffit d'indiquer après l'indicateur du DOS.

```
A>DOSDISK
A>
A>REM CREATION D'UNE DISQUETTE MSX-DOS
A>REM -----
A>
A>REM ATTENTION REPONDEZ BIEN A LA QUESTION SUIVANTE
A>
A>VERIFY ON
A>FORMAT
Drive name? A,B: B
1 - Single side,,,
2 - Double side,,,
2
Strike a key when ready,,,
Format complete
A>COPY MSXDOS.SYS B:
1 Files copied
A>COPY COMMAND.COM B:
1 Files copied
A>
```

Si vous entendez CTRL-C durant l'exécution d'un fichier les commandes, la station prendra la commande suivante :

```
Terminal batch file A:\N1?
Si vous tapez Y, le système arrêtera l'exécution du fichier de commande et l'indicateur du DOS ré-apparaîtra. Si vous appuyez N, la commande qui a été interrompue sera le CTRL-C sera insérée à la suite des commandes du fichier s'exécutant. Il n'est donc pas conseillé d'interrompre un fichier de commande par CTRL-C, la commande PAUSE a été spécialement créée à cet effet comme vous le voyez l'exemple suivant :
```

```
A>COPY CON DELETE.BAT
REM PROGRAMME D'EFFACEMENT DES FICHIERS .TXT
REM -----
```

```
A>COPY CON DELETE.BAT
REM PROGRAMME D'EFFACEMENT DES FICHIERS .TXT
REM -----
```


0.2.3 Les arguments d'un fichier de commande

Tout que nous l'avons vu jusqu'à présent, l'inconvénient du fichier de commande est qu'il exécute une tâche bien précise sans aucune possibilité de variante. Ainsi, dans l'exemple suivant, le fichier de commande recopie tous les fichiers .TST en .TXT et efface les fichiers .TST d'origine.

```
A>COPY EOM TEST.TST,BAT
COPY *.TST *.TXT
DEL *.TST
^Z
```

Si nous voulions copier les fichiers .AND vers .TST et effacer les fichiers .AND d'origine, il aurait fallu créer un autre fichier de commandes.

Grâce aux arguments du fichier de commande, nous allons voir qu'il est possible de créer un seul fichier de commandes et lui faire exécuter des tâches différentes selon que statisme. Regardons d'abord l'exemple de création suivant.

```
A>COPY COM EOPDEL,BAT
```

```
COPY %1 %1
DEL %1
^Z
```

Nous reconnaissons immédiatement la commande COPY vers la disquette. Si elle n'a pas été créée, il ne s'agit pas ici de copier le fichier qui porte le nom %1. Le symbole %1 représente en quelque sorte une variable dont la contenu va être déterminé au moment de son appel. Ainsi, si nous posons :

```
A>COPDEL TEST,BAT
```

EOPDEL est le nom du fichier de commande et TEST.BAT est appelé le premier argument de la commande. En premier argument va être introduit dans la variable %1 et dès lors :

```
COPY %1 %1      sera interprété en      COPY TEST,BAT %1
DEL %1          sera interprété en      DEL TEST,BAT
```

ce qui provoquera la copie du fichier TEST.BAT vers la disquette et l'effacement du fichier TEST.BAT sur la disquette par défaut.

```
A>COPDEL *.TST
```

provoquera la copie de tous les fichiers .TST vers la disquette et l'effacement de ces mêmes fichiers du disque par défaut.

Le symbole %1 représente donc une variable qui sera remplie par le premier argument de la commande. Mais, le système ne peut pas plus, car il existe un symbole pour 9 variables dont la contenu prendra des 9 arguments maximum que l'on peut ajouter au nom du batch étendu.

```
A>DATEHEUR %1-%2-%3
```

```
DATE %1-%2-%3
TIME %1:%2:%3
^Z
```

Nous voyons ici que 6 variables ont été utilisées (%1 à %6). %1 doit recevoir le jour du mois, %2 le mois, %3 l'année, %4 les heures, %5 les minutes et %6 les secondes. Le premier argument qui suivra l'appel DATEHEUR.BAT sera affecté à la variable %1, le deuxième argument à la variable %2, et ainsi de suite jusqu'à %6.

```
A>DATEHEUR 21 9 1987 14 15 10
```

```
A>
A>DATE 21-09-1987
A>TIME 14:15:10
A>
```

Par l'exemple ci-dessus, on voit que les arguments doivent être séparés par un espace, qu'ils sont affectés aux variables %1 à %6 dans l'ordre des positions de ces arguments dans la ligne de commande et que les commandes DATE et TIME sont affectées au moment de leur exécution avec la contenu des variables plutôt qu'avec le nom de ces variables.

Autre exemple :

```
A>COPY COM SUPERCOPY.BAT
```

```
TYPE %1
REN EST-CE BIEN LE FICHIER QUE VOUS VOLEZ COPIER ?
PAUSE OUJ -> RETURN, NON -> CTRL-C
REN SOUS LE NOM : %2 ?
PAUSE OUJ -> RETURN, NON -> CTRL-C
COPY %1 %2
REN FAUT-IL EFFACER %1 ?
PAUSE OUJ -> RETURN, NON -> CTRL-C
DEL %1
^Z
A>
```

Exécution :

```
A>SUPERCOPY A1TOTO.TST B1TEXTE.TXT
```

```
A>TYPE *.TOTO.TST
Petit fichier d'essai de deux lignes.
Eci est la deuxième ligne de TOTO.TST
```

```
A>REN EST-CE BIEN LE FICHIER QUE VOUS VOLEZ COPIER ?
A>PAUSE Out -> RETURN, NON -> CTRL-C
Strike a key when ready...
A>REN SOUS LE NOM : B1TEXTE.TXT ?
A>PAUSE OUJ -> RETURN, NON -> CTRL-C
Strike a key when ready...
A>COPY A1TOTO.TST B1TEXTE.TXT
A>REN FAUT-IL EFFACER A1TOTO.TST ?
A>PAUSE Out -> RETURN, NON -> CTRL-C
Strike a key when ready...
A>DEL A1TOTO.TST
```


1. LES DISQUES DE 3 1/2"

Il existe enfin une dixième variable appelée X0 dont l'emploi est facultatif et qui reçoit toujours la même contenu à savoir le nom même de votre fichier de commande. Ainsi dans l'exemple précédent on aurait pu ajouter au début du fichier de commande les deux lignes de Remarques suivantes :

```
REN X0 : PROGRAMME DE COPIE CONVIVAL
REM
```

et à l'exécution, cela aurait donné :

```
A>REN SUPERCOP : PROGRAMME DE COPIE CONVIVAL
A>REN
etc...
```

Une dernière remarque. Si vous employez la variable X3, par exemple, la ligne de commandes que vous devez poser pour appeler le fichier de commandes doit contenir 3 arguments obligatoirement. Sinon, le système ne trouve pas de troisième argument produira le message "File not found".

```
A>COPY CON TEST.BAI
A>TYPE X3
^Z

A>TEST TEXTE.TXT
A>TYPE
File not found
A>
```

Par contre, la méthode suivante marchera car "AA" servira de premier argument, "BB" de deuxième et "TEXTE.TXT" sera donc bien le troisième argument :

```
A>TEST AA BB TEXTE.TXT
A>TYPE TEXTE.TXT
contenu du fichier TEXTE.TXT
A>
```

	TYPES DE DISQUE			
	FB	F9	FA	FB
Media descriptor				
Nombre de Faces	1	2	1	2
Nombre de Pistes	80	80	80	80
Nombre de secteurs par piste	9	9	8	8
Taille du secteur	512	512	512	512
1 ^{er} Secteur FAT	1	1	1	1
Taille de la FAT	1024	1536	512	1024
Nombre de FAT	2	2	2	2
1 ^{er} Secteur de la Directory	8	7	3	5
Nombre de secteurs dans la Directory	7	7	7	7
Nombre d'entrées par secteur Directory	16	16	16	16
Nombre maximum d'entrées Directory	112	112	112	112
1 ^{er} secteur pour fichier	12	14	10	12
Taille du cluster	1024	1024	1024	1024
Nombre de cluster pour fichiers	354	713	315	634
Nombre de secteurs total	720	1440	640	1280
Capacité utilisateur	362496	730112	322560	649216
Capacité formatée	368640	737280	327680	655360

2. LES DISQUES DE 5K

Media descriptor	TYPES DE DISQUE			
	FD	FD	FE	FF
Nombre de Faces	1	2	1	2
Nombre de Pistes	40	40	40	40
Nombre de secteurs par piste	9	9	8	8
Taille du secteur	512	512	512	512
1er Secteur FAT	1	1	1	1
Taille de la FAT	1024	1024	512	512
Nombre de FAT	2	2	2	2
1er Secteur de la Directory	5	5	3	3
Nombre de secteurs dans la Directory	4	7	4	7
Nombre d'entrées par secteur Directory	16	16	16	16
Nombre maximum d'entrées Directory	64	112	64	112
1er secteur pour fichier	9	12	7	10
Taille du cluster	512	1024	512	1024
Nombre de cluster pour fichiers	331	354	313	315
Nombre de secteurs total	360	720	320	640
Capacité utilisable	179712	362496	160256	322560
Capacité formatée	184320	368640	163840	327680

LE LIVRE DU DISQUE MSX

CE LIVRE CONVIENT AUX MSX I ET II
DE TOUTES MARQUES

Dans ce livre, vous découvrirez tout sur la structure et l'organisation des disquettes MSX quelle qu'en soit la marque, la taille ou la capacité, CLUSTER, RECORD, BOOT, FAT, DIRECTORY, rien ne manque. Vous y trouverez aussi toutes les adresses et tous les points d'entrée de toutes les routines des extensions disque. Chaque routine est accompagnée d'un exemple d'utilisation et de programmation en Assembleur.

Un chapitre complet est réservé au DISK BASIC, à ses commandes et à la manipulation de tous les types de fichiers. Chaque explication est agrémentée de nombreux exemples.

Un autre chapitre est réservé à la description détaillée de l'éditeur du MSXDOS et à la structure des fichiers BATCH.

Cet ouvrage est le plus bel OUTIL DE REFERENCE écrit à ce jour sur les disques des systèmes MSX I ou II.

BCM s.c.

24, route de la Sapinière - B-4960 Banneux Belgique

ISBN 2-87111-010-7



110 FF